Commodore

Cena 12 tys. zł.
nr indeksu 355275

9 '92

KEFIR

Miesięcznik Użytkowników Komputerów C-64 i Amiga



Commodore



nr indeksu 355275

Wydawca:
KEBAB - sp. z o. o.
ul. Wojciechowskiego 28
PL - 71-746 Szczecin
telefon: (091)77674
fax: (091)45402

Redaguje kolegium
w składzie:
Krzysztof Kobus
Patryk Łogiewa
Grzegorz Mikuła
Krzysztof Moroń
Mirosław Smyk
Paweł Sołtysiński

Redaktor naczelny:
Patryk Łogiewa

Szef działu AMIGA:
Krzysztof Kobus
tel. (091)525336

Szef działu C-64:
Paweł Sołtysiński
tel. (091)77674

Redakcja nie zwraca nie zamówionych materiałów oraz zastrzega sobie prawo wprowadzania zmian w otrzymanych rękopisach.

Wydawca nie odpowiada za treść zamieszczanych ogłoszeń.

Projekt okładki:
Tomasz Kuczyński

Commodore



PRENUMERATA

Każdy egzemplarz zakupiony bezpośrednio u nas kosztuje odpowiednio:
numery do 6'92 włącznie - 9,5 tys. zł.
numery 7-8'92 i następne - 11 tys. zł.
Oznacza to , że można zamawiać numery zaległe jak też zaprenumerować jeszcze nie wydane. odbywa się to tylko poprzez dokonanie odpowiedniej wpłaty na nasze konto.
Na odwrocie każdego odcinka kuponu wpłaty należy dokładnie napisać, których numerów wpłata dotyczy.
W przypadku prenumeraty można zamawiać numery d0 końca aktualnego okresu "małej premumeraty" obejmującej zawsze trzy numery. Przypominamy więc jeszcze raz aktualna cena egzemplarza wynosi: **11.000,- zł.**

Nasze konto:
Pomorski Bank Kredytowy
II Oddział w Szczecinie
numer konta: 368113-25771-136

Podajcie dokładny adres, imię i nazwisko zamawiającego oraz oraz numery egzemplarzy, których wpłata dotyczy na odwrocie każdego z odcinków blankietu wpłaty.

Pomimo naszych ciągłych próśb o dokładne wypełnianie kuponów wciąż otzymujemy wpłaty, z którymi nie wiemy co zrobić. Prosimy więc przy okazji o podanie dokładnego adresu następujące osoby:

BABULA EWA z Leszna
IGNOCIUK ANDRZEJ z Poznania
JARMOSZEWICZ JAN z Gryfic
KUCZKOWSKI STEFAN z Piły

I ostatnia uwaga - **wszelką korespondencję, wpłaty etc. kierujcie tylko i wyłącznie na adres redakcji**

REKLAMA

Ogłoszenia drobne od osób indywidualnych (do 10 słów na wyciętym kuponie z III-ciej strony okładki) przyjmujemy bezpłatnie.
Ogłoszenia drobne od osób prawnych oraz zawierających powyżej 10 słów - 1000 zł. za słowo.
Ogłoszenia ramkowe (minimalny format 20 cm^2):
1 cm^2 - 4.500,- zł., cała strona 2,5 mln zł., cała IV strona okładki - 4 mln. zł., 1/2 tej strony - 2,5 mln. zł;
dodatkowy kolor - odpowiednio 50% drożej.
Ogłoszenia prosimy przesyłać listem poleconym.



Nr 9 Wrzesień 1992

Mini-penetrator
gra do własnoręcznego wpisania na C-64

AMOS cz. II
procedury, etykiety, lista instrukcji

High Speed Pascal 1.0
jeszcze raz okiem programisty

Amiga 4000 - nowy produkt firmy Commodore

A2024 - monitor wysokiej rozdzielczości
prawie wszystko o...

Spis treści:



02 Z kraju i ze świata
03 Superkomputery
05 Nowe booty dla BootX'a
06 AMOS cz.II
09 High Speed Pascal 1.0
11 Mapa pamięci Amigi c.d.
12 Reset? Czemu nie
14 Jak scrunchować demo?
15 NTSC dla każdego
16 Assembler dla C-64 - odcinek kolejny
19 INPUTER - nowy system publikacji programów
20 Mini-penetrator - krótka gra na C-64
21 Basic - dokończenie
22 Master Head - recenzja gry
23 BASIC - PROTECTOR czyli programy na zamówienie
24 COPY - PARTY regulaminy konkursów
26 Ankieta do czytelników
27 Amiga 4000 - nowy produkt firmy Commodore
28 Nowe gry na rynku wg Amiga World
30 A2024 - monitor wysokiej rozdzielczości
32 Ogłoszenia drobne
33 Listingi:
O przerwaniach (z numeru 7-8/92)
Nowe booty dla BootXa
Reset
NTSC
INPUTER
Mini-penetrator
Basic-protector
40 Kebab-Mon V.5
Od redakcji













Dobra wiadomość dla maniaków szybkości - procesor Motorola 68040 ma wreszcie wersję 33MHz, która na dodatek daje się taktować zegarem nawet do 35MHz. Natychmiast pojawiły się w sprzedaży akceleratory firmy *Progressive Peripherials & Software* wyposażone w to cudeńko, zdolne osiągnąć do 26-ciu MIPSów (milionów operacji na sekundę).

Dobrze jest gdy producent naszego ulubionego programu użytkowego dba o swoich klientów i nie zapomina o systematycznym publikowaniu nowych, poprawionych jego wersji. Gdy jednak firma jest zbyt powolna, ulegnie rozwiązaniu, lub po prostu straci zainteresowanie projektem, który jest jednak wart dalszego rozwijania, często zdarza się, iż koderzy ze sceny kontynuują prace swoich profesjonalnych kolegów - nierzadko z dużo lepszym skutkiem. Tak było na przykład w przypadku assemblerów SEKA i ASM-One, kilku archiwizerów, czy wszelakiej maści programów muzycznych i kopiujących. Na ogół wydawcy oryginału nie podnosili z tego powodu rwetesu, mimo iż częstokroć w ulepszonych wersjach nie figurowały nawet personalia autorów pierwowzoru i nazwa producenta. Gdy jednak rynek został dosłownie zasypany "klonami" *X-Copy* (zjawisko trudne do przeoczenia), które stworzono po to by ich "twórcy" ograniczający się najczęściej tylko do zmiany grafiki mogli zabłysnąć na scenie - firma będąca właścicielem praw do tego produktu - *Cachet* - podjęła zdecydowane kroki przeciwko takim praktykom. Jeden szczególnie domorosły programista, ukrywający się pod ksywą Uli Magen zalał firmie Cachet sadła za skórę, za co ta odwdzięczyła mu się ni mniej ni więcej tylko publikacją LISTU GOŃCZEGO oferującego 5000 marek osobie, która umożliwi jego ujęcie. Denuncjatorów oczywiście nie zabrakło i według najświeższych pogłosek obie zainteresowane strony (Magen i Cachet) są już w kontakcie. Najbliższe tygodnie pokażą, czy sprawa trafi do sądu...

Na miano największego "newsa" numeru zasługuje niewątpliwie karta *OpalVision* firmy *Centaur Development* - pierwszy produkt, który jest w stanie konkurować z VideoToasterem. Jak na razie wszystkie nasze informacje o nim pochodzą z reklam w zachodnich czasopismach, są one jednak na tyle interesujące, że warto sobie wypunktować co istotniejsze z nich, odkładając jednocześnie pełne omówienie urządzenia do czasu gdy dostaniemy je w swe ręce.

A oto krótki z konieczności opis *OpalVision*:

- 24 bity grafiki (16,8 mln kolorów), która może pojawić się za lub przed standardową grafiką Amigi,

- rozdzielczość do 768x480 (580 w PAL-u),

- współpraca z Amigą 500, 600, 2000 i 3000,

- koprocesor graficzny (technologia VLSI) i 1,5 MB display RAM-u,

- automatyczna samokonfiguracja dla NTSC i PAL-u.

Tu przerwę na moment wyliczankę aby wspomnieć, że karta charakteryzuje się modularną konstrukcją, pozwalającą na rozbudowę wedle potrzeb i zasobności kieszeni. Powyższa lista obejmuje zasoby płyty głównej, na której można instalować co następuje:

Frame Grabber/Genlock Module:

- możliwość przechwycenia pojedyńczej, kolorowej ramki z sygnału "live",

- pełna integracja "live" wideo z grafiką 24-bitową za pomocą tzw. kanału alfa, pozwalającego na ustalenie 256 poziomów przezroczystości,

Scan-Rate Converter:

- doskonałej jakości flickerfixer z możliwością wykorzystania jako niezależny, 24-bitowy bufor dla grafiki,

Quad-input Production Switcher:

- dodatkowa jednostka dołączana do modułu Frame Grabber/Genlock umożliwiająca programową obróbkę (przejścia, miksowanie) dwóch niezależnych sygnałów video,

- zawiera cztery wejścia wizyjne z których każde może być przyporządkowane jednemu z dwóch kanałów (A, B)

- umożliwia wszystkie podstawowe efekty "przejść" pomiędzy dwoma sygnałami

- wszystkie wejścia oraz wyjście pracują z sygnałami S-Video (S-VHS, Hi-8) oraz Composite Video.

Opal Vision Roaster Chip:

- cyfrowe przetwarzanie i obróbka wchodzącego sygnału wizyjnego w czsie rzeczywistym.

- efekty typu odwracanie, skalowanie, owijanie na kuli itp. dowolnego (ruchomego) obrazu z dowolnego wejścia

- PIP czyli podgląd sygnału wizyjnego w "normalnym", skalowalnym okienku Workbench'a w pełnej (!) 24-bitowej skali kolorów.

Do systemu Opal Vision dołączane jest specjalne oprogramowanie. Oprócz pakietu do "malowania" o nazwie *Opal Paint* otrzymujemy pakiet prezentacyjny *Opal Presents*. *OpalVision Hotkey* to moduł umożliwiający wywołanie za pomocą kombinacji klawiszy lub komend Arexx'a ekranu 24-o bitowego w dowolnym momencie. Oprócz tego na posmaczek otrzymujemy pierwszą na świecie grę z grafiką wykorzystującą paletę 16,8 mln kolorów. Gra ta nosi tytuł *King of Karate*.

Z nie potwierdzonych źródeł wiemy o planowanej przez firmę Commodore na początek przyszłego roku, prezentacji nowej Amigi oznakowanej numerem 4000. Owa maszyna zaopatrzona ma być w procesor MOTOROLA 68040 oraz nowe układy obsługujące dźwięk i grafikę nazwane roboczo *AA chip set*. Dzięki zastosowaniu wspomnianych "kości" użytkownicy Amigi 4000 będą mieli możliwość wyświetlania obrazu w rozdzielczościach podobnych jak w przypadku ECS, z dodatkiem trybów 800x300, 800x600 (Interlace), 1280x480, 1280x960 (Interlace). Powyższe tryby umożliwiają jednoczesne wyświetlenie 256 kolorów zapisanych na 8 Bit-Plane'ach wybranych z 24-bitowej palety barw (ponad 16 milonów kolorów), lub wyświetlenie trybu HAM opartego na 6 Bit-Plane'ach.

O wiele większe zmiany dotyczą jednak możliwości dźwiękowych komputera. I tak będzie można używać czterech niezależnych 16-to bitowych kanałów Audio pracujących z częstotliwością 56kHz, mogących emulować osiem kanałów 8-mio bitowych na częstotliwości 56kHz, lub szesnaście 8-mio bitowych kanałów przy częstotliwości 28kHz.

Jakość dźwięku uzyskana przy pomocy takich układów nie ustępuje popularnym odtwarzaczom płyt kompaktowych renomowanych firm światowych. Ponadto nowa Amiga wyposażona ma zostać w twardy dysk o pojemności 105 Mb, znane z A3000 złącze Zorro III, 16 Mb pamięci FAST, 4Mb pamięci CHIP, które to mogą zostać rozszerzone odpowiednio do 64Mb i 16Mb.

Znana m.in. z karty graficznej *Visiona* firma *Xpert Computer Services*, wprowadza aktualnie na rynek swój nowy produkt o nazwie *Domino*. Jest to karta graficzna umożliwiająca wyświetlenie obrazu z częstotliwością 72Hz (Flicker-Fixer). Oprócz tego karta potrafi generować obraz o następujących rozdzielczościach:

800x600 (32768 kol.)

1024x768 (256 kol.)

1120x832 (256 kol.)

Aby jednak można było w ogóle uruchomić kartę, musimy posiadać Amigę z procesorem co najmmniej 68020 oraz Kickstart 2.0. Wszystkie programy, które nie wykorzystują więcej niż 16 kolorów i są zaprogramowane "czysto" tzn. trzymają się zaleceń twórców systemu OS 2.0, można natychmiast uruchomić w wysokiej rozdzielczości na karcie *Domino*. Oprócz tego karta jest w dużym stopniu kompatybilna ze wspomnianą już kartą *Visiona*. Np. program TV-Paint Visiona może również współpracować z nowym produktem firmy. Cena promocyjna karty została ustalona na 698,- DM

Nowe rzeczy dzieją się także w dziedzinie obróbki dźwięku na Amidze. Coraz częściej pojawiają się produkty umożliwiające odczyt, obróbkę i ponowny zapis dźwięku w postaci 16-o bitowych sampli.

Firma *Advanced Systems & Software* proponuje w tej dziedzinie kartę o nazwie *AD516*. Posiada ona następujące możliwości: 16-o bitowy sampling, 64-o krotny oversampling i częstotliwość próbkowania do 50KHz. Zapis może oczywiście odbywać się w czasie rzeczywistym pod warunkiem posiadania szybkiego (szybszego niż np. A590) twardego dysku, cena na razie ok. 3000,- DM.

Nieco tańszym rozwiązaniem tego typu wydaje się być prezentowana przez firmę *Macro Sytem* karta o wdzięcznej nazwie *Maestro Proffesional*. Następczyni znanej już od prawie roku karty *Maestro* daje nam również możliwość samplowania z 16-o bitową rozdzielczością oraz częstotliwościami 32, 44.1, i 48 KHz. Optyczne oraz elektryczne wejścia cyfrowe umożliwiają podłączenie odpowiedniego (cyfrowego) źródła sygnału. W połączeniu z magnetofonem typu DAT można również samplować ze źródła analogowego (np. mikrofonu).

Wśród wielu możliwości obróbki sygnału zarówno zapisanego już na dysku jak i bezpośrednio doprowadzonego do karty z zewnątrz, ciekawą dla naszych piratów może się okazać filtrowanie tzw. bitów SCMS (zabezpieczenie uniemożliwiające bezpośrednie przekopiowanie zapisu z jednego magnetofonu DAT na drugi).

To oczywiście jak na razie tylko żart. Dopóki w każdym (lub prawie każdym) domu nie będzie stał magnetofon DAT, twórcy cyfrowych zapisów na taśmach magnetycznych mogą spać spokojnie. Piratom się po prostu nie opłaci...

Również znany program prezentacyjny *Scala* doczekał się starszego brata. *Scala MM200* tak nazywa się nowy pakiet. *MM* powinno oczywiście kojarzyć nam się z aktualnym magicznym hasłem rynku komputerowego - Multimedia. 135 rozmaitych efektów oraz możliwość sterowania urządzeniami zewnętrznymi takimi jak laserowe odtwarzacze video-płyt (Laserdisc), magnetowidy profesjonalne, CDTV, MIDI i wiele innych, stawia ten pakiet istotnie wysoko w notowaniach. cena? tylko (?) 999,- DM

Prawdziwy szok przygotowała nam firma Atari. Od dawna krążyły co prawda pogłoski, że szykuje ona poważnego konkurenta dla Amigi ale teraz już wiemy czego się bać (?). Produkt ten o dumnie brzmiącej nazwie *Atari Falcon 030* wygląda z zewnątrz jak zwyczajne ST. W środku natomiast mamy kilka ciekawostek. M.in. zainstalowany, podobnie jak w A600HD, twardy dysk 2,5 cala. Oprócz tego (nie tak jak w A600) procesor 68030, 16MHz blitter, procesor graficzny umożliwiający uzyskanie 32768 kolorów na ekranie w rozdzielczości 320x200, układ obróbki dźwięku pozwalający na uzyskanie jskości CD (16 bitów, 50KHz)! oraz specjalny układ znany m.in. z NeXT'a o nazwie Digital Signal Processor DSP-56001. Sam ten układ potrafi osiągnąć prędkosć ok. 16 MIPS'ów przy 32MHz. Do kompletu dołączony jest jeszcze wewnętrzny drive FD-HD 3.5 cala/1.44MB...

Jednym zdaniem: Nowy komputer domowy ale nie w stylu firmy Commodore (patrz A600). I co na to Commodore?



Superkomputery

Z czasem każdy użytkownik komputera, niezależnie od posiadanego typu i modelu dochodzi zgodnie z którymś tam prawem Murphy'ego do wniosku iż:

- jego komputer posiada zbyt mało pamięci operacyjnej;
- możliwości obliczeniowe mimo iż sięgnęły zenitu - są niewystarczające.

Zdarzają się także i inne kłopoty, lecz w zasadzie te dwie sprawy spędzają sen z powiek wielu komputerowcom. Tak więc ci, których na to stać rozszerzają pamięć w swoich "maszynach", dokupują "dopalacze", by w końcu osiągnąć prędkość... kilkanastu MIPS'ów i... na tym koniec!!! Dla wielu takie udogodnienia stanowią szczyt marzeń, ale co z tymi, których to nie zadowala? Co zrobić, gdy na rozwiązania problemów stawianych przed komputerem trzeba czekać wieki, a czas to pieniądz, jak mówi przysłowie...

W tej sytuacji oznacza to zapewne fakt, iż słowo "superkomputer" nabierze dla niego realnego znaczenia.

W tym miejscu nie sugeruję oczywiście nikomu, by zazdroszcząc na przykład *NASA*, wstawił sobie do domu elektroniczne cacko wielkości szafki kuchennej, ale... kto wie? Może to nie byłby głupi pomysł???

Problem może stanowić jedynie cena, która stanowi liczbę złożoną z kilkunastu cyfr.

Tak oto dotarliśmy do właściwego tematu artykułu...

O superkomputerach słyszał chyba każdy, mający cokolwiek wspólnego z informatyką bądź naukami pokrewnymi, który poważnie zajął się obróbką i przetwarzaniem danych. Stanowią one niezbędne narzędzie pracy uczonych, wojskowych i wszystkich tych, którym potrzebne są ogromne ilości przeanalizowanych danych (meteorolodzy, sejsmolodzy, etc.).

Jak doszło do tego, iż dziś istnieje tak doskonały sprzęt???

By odpowiedzieć na to pytanie - należy cofnąć się w przeszłość, do roku 1976. Wtedy to właśnie zamiast starego, wysłużonego *CDC 7600* pojawił się legendarny już dziś *Cray-1*. Oba komputery zostały zaprojektowane przez Seymour'a Cray'a, jednakże mimo wielu podobieństw między obiema jednostkami *Cray-1* wyróżniał się licznymi usprawnieniami technicznymi i stanowił duży krok w stosunku do swego poprzednika. Prace nad stworzeniem systemu rozpoczęły się jednak dużo, dużo wcześniej i nie zostały zakończone z chwilą stworzenia całego urządzenia, a wręcz przeciwnie - trwały nadal, a ich owocem stał się *Cray-2*.

O tym komputerze powiadomiono już w czerwcu 1985 roku, jednakże ostateczny projekt wyglądu i wnętrza jednostki przypadł na ostatnią dekadę tegoż roku. Próby i testy praktyczne pozwoliły ponownie przemyśleć architekturę opartą na równoległej pracy kilku procesorów i umożliwiły ostateczne wyciągnięcie wniosków.

Cray-2 to architektura bardzo zwarta; składająca się z 14 pionowych kolumn, rozmieszczonych tak, iż całość z góry - przypomina kształtem literę 'C'. Jego średnica wynosi 135 cm, 115 cm - stanowi wysokość, całość zawiera liczne "kości", wykonane w technice ECL (Emitter Coupled Logic) oraz MOS (Metal Oxide Semiconductor), zanurzone w obojętnej chemicznie cieczy CF_4, stanowiącej substancję chłodzącą.

Nietrudno zgadnąć, iż kolejny krok naprzód w technologii rodziny *Cray'ów* stanowi model o numerze '3'.

W chwili obecnej prace nad jego konstrukcją są już zapewne skończone, gdyż przewidywany pokaz miał przypaść na początek lat 90-tych.

W maju 1988 roku w laboratorium testowano prototyp, oparty na technice wykorzystującej GaAs (arsenek galu) w produkcji układów scalonych... Co to dało??? Nowa technologia to zazwyczaj dalsza miniaturyzacja; tym razem rzut z góry stanowi ośmiokąt foremny, w przekroju o wymiarach tylko 80x85 cm.

Najbardziej jednak ciekawią dane techniczne obu superkomputerów...

	Cray-2	Cray-3
Moc:	150 kVA	150 kVA
Chłodzenie:	CF4	CF4
Kształt:	litery C	ośmiokątny
Wysokość [cm]:	115	85
Średnica [cm]:	135	80
Chip:	ECL	GaAs
Gate/chip:	16	400
Wymiary płyty [cm]:	20x10x0.09	2.5x2.5x0.04
Ilość warstw/płytę:	6	8
Ilość płyt/moduł:	8	64
Ilość modułów/system:	320	200
Liczba procesorów:	4	16
Wykonywanie rozkazów [MIPS]:	500	8000
Wielkość pamięci [Mwords]:	128-512	128-2096
Okres taktu zegara [ns]:	4.1	2
Moc obliczeniowa [GFLOPS]:*	1.8	16

Jak widać z tabeli możliwości są szokujące. W tym miejscu ktoś może zgryźliwie zauważyć, iż możliwości są adekwatne do kosztów i... będzie miał rację. Sprzętu tej klasy niewiele jest na świecie.

Od 1985 roku liczba systemów komputerowych, w cenie rzędu 100000$ - 1000000$ i szybkościach 10-200 [MFLOPS] wzrosła zdecydowanie do ok. 50 rodzajów, co stanowiło, po pierwsze, wypełnienie dotkliwej "luki" między sprzętem tanim, ale za to o małych mocach obliczeniowych, a superkomputerami w cenie powyżej 10 milionów dolarów, po drugie przyciągnęło uwagę naukowców, inżynierów i innych ludzi o pokrewnych profesjach, gdyż maszyny te były stosunkowo "niedrogie", zaś możliwościami starały się dorównywać profesjonalnym, unikalnym jednostkom.

Te urządzenia, zwane *"Crayettes"* znalazły zastosowanie zarówno w środowisku naukowym, jak i poza tym kręgiem. O stałym wzroście popularności niech świadczą poniższe liczby:

Kraj	Liczba superkomputerów wykorzystanych w	
	pracy naukowej	innych dziedz. życia
USA	17	110
Japonia	15	58
Niemcy	6	15
Wielka Brytania	4	15
Francja	3	12
Włochy	2	0
Australia	2	1
Holandia	1	1

Liczba komputerów o słabszych mocach obliczeniowych,

Kraj	wykorzystywanych w:	
	pracy naukowej	innych dziedz. życia
USA	5795	24910
Japonia	3538	11632
Niemcy	1713	2795
Wielka Brytania	780	3250
Francja	1246	1990
Włochy	525	0
Australia	267	70
Holandia	200	80

Powyższe dane pochodzą z października 1987 roku, Polska w tej statystyce niestety nie występuje...

Pozostaje jeszcze odpowiedzieć na pytanie kto ponadto wykorzystuje te superszybkie jednostki. Uważni Czytelnicy "Kebaba" pamiętają zapewne artykuł kolegi Miłka Smyka z numeru 6 tego pisma, poświęcony technice i efektom specjalnym współczesnej kinematografii. Komputery na pewno stanowią ułatwienie w kreowaniu scen, ale trzeba wziąć pod uwagę fakt, iż efekt wyjściowy musi idealnie pasować do scen kręconych tradycyjnie, wymagana jest więc tu wysoka wierność i jakość obrazu syntetycznego. O trudnościach, z jakimi boryka się reżyser filmowy niech świadczy następujący przykład:

Jeżeli jedna "ramka" filmu złożona będzie z 20 milionów pikseli (biorąc pod uwagę film kolorowy), a na każdy piksel przypadnie tylko 100 instrukcji maszynowych, to w sumie na jeden raster będziemy musieli wykonać ok. 2 miliardów rozkazów!!!

Cray'owi z serii *X-MP2*, wykorzystującemu oba procesory przetworzenie takiej scenki zajęłoby ok. 10 sekund. Wygenerowanie bardziej złożonych obrazów zajęłoby około 60 razy więcej czasu. A gdyby tak zastosować wolniejszy komputer, powiedzmy *VAX 11/780*??? Czas potrzebny na przetworzenie danych ekstremalnie - mógłby sięgnąć 10 godzin!!!

Aby jeszcze bardziej przybliżyć Czytelnikowi złożoność problemu dodam, iż wykonanie minuty takiego filmu - zajęłoby *"Cray'owi..."* ok.10 dni, podczas, gdy dla *VAX'a* - oznaczałoby to 3 lata pracy.

Ujęte tu dane potraktować należy jednak z "przymrużeniem oka", gdyż odstępstwa od reguły zdarzają się wszędzie (także i w tym przypadku), a jako kontrprzykład można wymienić tu choćby "Terminatora 2"... Przetwarzanie obrazów stanowi więc swego rodzaju sztukę gdzie niezbędna jest spora wiedza z zakresu wyższej matematyki i umiejętność tworzenia odpowiednich (czytaj: działających sprawnie) algorytmów, a co za tym idzie także programów. Technika zwana ray-tracing'iem (znana także użytkownikom Amig) jest zapewne efektowna, ale po pierwsze wymaga dużej cierpliwości ze strony użytkownika (bądź szybkiego sprzętu), po drugie - konieczne jest posiadanie pamięci operacyjnej o dużych rozmiarach. Skomplikowane sceny do filmu *The Last Starfighter*, złożone z ponad miliona wielokątów, zajęły, stosując technikę tworzenia syntetycznych obrazów lata...

Inne zadania, do rozwiązywania których stosowane są komputery - to opracowywanie map pogody, analiza danych na temat aktywności skorupy ziemskiej, symulowanie złożonych procesów fizyko-chemicznych, projektowanie i testowanie nowych maszyn i urządzeń (np.: *The Boening Airplane Company* - modyfikując model 737 - zamiast tunelu aerodynamicznego - wykorzystała komputer). Nie sposób pominąć tu techniki kosmicznej i zastosowania militarne, a także wiele, wiele innych dziedzin naszego życia. Komputery stanowią nieodzowne narzędzie pracy, pojawiają się dosłownie wszędzie: w domach towarowych, na dworcach, w kasach, biurach, wszędzie tam, gdzie niezbędna jest obróbka dużych ilości informacji. Ułatwiają życie, a czasami nawet ratują je. Stanowią nieodzowny element środowiska XX wieku, osiągają, podobnie jak człowiek kolejne szczeble rozwoju.

Mam nadzieję, iż już wkrótce nawet tak duże systemy, które chciałem nieco przybliżyć Czytelnikom, staną się bardziej dostępne dla zwykłego "śmiertelnika", a póki co nie pozostaje mi nic innego, jak zakończyć ten artykuł parafrazą sentencji: "Per computer ad astra... "

Krzysztof "Brain" Franckowski.

W artykule wykorzystano wiadomości zawarte w książce: "Supercomputers and their Use" - Christopher'a Lazou.

* [GFLOPS] Giga operacji zmiennoprzecinkowych na sekundę...

Nowe booty dla BootX'a

Programów do wykrywania i usuwania wirusów wydaje się ostatnimi czasy powstawać więcej niż paskudztwa z którym mają walczyć. Od najprostszych "bootblock protectors" (łatwo zresztą dających się ogłupić), poprzez programy dedykowane zabijaniu określonego wirusa - tu prym wiodą *Bush*, *Schwarzkopf* i inne *Saddam-killer'y*, do skomplikowanych "kombajnów" antywirusowych takich jak *Virus Expert* czy *BootX*.

I właśnie ostatni z tych programów, dzieło Petera Stuera - członka stowarzyszenia SAFE HEX INTERNATIONAL, przyciągnął mą uwagę na dłużej. Wysoka efektywność, estetyczne wykonanie, przemyślana konstrukcja i co w tej "branży" istotne - regularnie - nawet co parę tygodni - ukazujące się nowe wersje sprawiają, że zdobywa on rzesze zwolenników, zarówno wśród "szarych" użytkowników jak i profesjonalistów. Jednakże nie opis tego doskonałego narzędzia będzie treścią poniższego artykułu - po ów odsyłam Was do któregoś z następnych numerów Kebaba. Tutaj zaś znajdą coś dla siebie osoby obeznane już z obsługą "BootX'a", pragnące nieco zwiększyć jego możliwości. Do dzieła zatem!

Bibliotekę bootblocków rozpoznawanych przez program można łatwo rozszerzyć wykorzystując funkcje *Learn* i *Save Brainfile*, co pozwala uniknąć uciążliwego bombardowania ekranu komunikatami o rzekomych wirusach w przypadku używania niestandardowych (na przykład samodzielnie napisanych) bootblocków. Autor, kierując się zapewne obawą przed wykorzystaniem jego programu do jakichś niecnych celów nie dał nam jednak możliwości instalowania dowolnych bootingów na dyskietkach, ograniczając możliwość wyboru do zasobów biblioteki BBLib. Osobiście uważam to za lekką przesadę, szczególnie w sytuacji gdy opcję taką oferuje konkurencyjny *Virus Expert*, choć więc pan Stuer wspominał w dokumentacji do "BootX'a" o mających się wkrótce pojawić nowych BBLib, postanowiłem z gruntu uznać wszelki oferowany asortyment za niewystarczający i stworzyć własną bibliotekę.

Zadanie okazało się bajecznie wręcz proste, w czym niemała zasługa "wewnętrznej" estetyki programu czyniącej jego analizę "lekką, łatwą i przyjemną" - wystarczyło zaledwie kilka minut harców D-Mon'em by znaleźć wszystko co potrzebne!

I tak: primo - jeśli jeszcze ktoś nie zauważył to plik BootX.BBLib jest plikiem wykonywalnym, w związku z czym można go skompresować *PowerPackerem* z aktywną opcją *LoadSeg*. A warto, bo składające się nań bootingi są zapisane w pełnej. jednokilobajtowej postaci oznaczającej w praktyce sporą liczbę zbędnych zer.

Secundo (wynika z primo) - bibliotekę najwygodniej będzie utworzyć korzystając z pomocy któregoś z asemblerów, jako że ten automatycznie wygeneruje odpowiednie hunki i ich nagłowki, zwalniając tym samym nasze głowy od tego niewdzięcznego zajęcia - zdecydowałem się na ASM-One'a.

I tercjo - format pliku:

- cztery bajty "BXBB" służące jako identyfikator zbioru,

- numer wersji programu przemnożony przez sto, zapisany na długim słowie (BootX 4.50 którego analizowałem sprawdza czy nie jest on niższy od 340, co sugeruje zmianę formatu biblioteki od wersji 3.40 właśnie),

- ilość bootblocków w pliku zapisana na długim słowie,

- jedno długie słowo z wartością aktualnie nieużywaną - aby jednak zachować kompatybilność z ewentualnymi kolejnymi wersjami należy tu umieścić kopię poprzedniej danej,

- numer bootblocku, którego nazwa jako pierwsza pojawi się w okienku BootX'a - również długie słowo,

a następnie właściwa zawartość biblioteki w formacie:

- trzydzieści bajtów identyfikatora (krótsze nazwy uzupełniane są zerami do tej właśnie długości),

- 1024 bajty samego bootblocku,

przy czym ostatnie dwa punkty struktury powtarzają się tyle razy ile zadeklarowaliśmy w pozycji trzeciej.

Uzbrojeni w powyższe informacje i program z **listingu 1**, możemy już przystąpić do pracy. Pierwszym krokiem będzie umieszczenie wszystkich interesujących nas bootingów w postaci plików na RAM-dysku.

Dla Czytelników, którzy nie wiedzą jak tego dokonać przygotowałem **listing 2**. Wystarczy "wklepać" go na ASM-One'a, włożyć do stacji dyskietkę, której bootblock ma stać się elementem naszej biblioteki i opcją "A" uruchomić asemblację. Po chwili pojawi się requester, gdzie najpierw aktywujemy urządzenie RAM, a następnie podajemy nazwę dla naszego pliku. Powyższą operację powtarzamy dla kolejnych bootingów.

UWAGA: Dysk, z którego uruchamiamy system musi posiadać "Ram-handler" w katalogu "L"!

Przebrnąwszy szczęśliwie przez powyższe zabiegi wpisujemy na ASM-One'a listing 1 i modyfikujemy jego ostatnią część, wpisując w kolejnych liniach:

```
Boot id,name
```

gdzie `id` to identyfikator, którego będzie używał BootX, a `name` to nazwa jaką ochrzciliśmy plik przy zapisie na RAM-dysk.

Potem już tylko asemblacja i zapisanie gotowej biblioteki na dysk komendą "WO". Jeśli nazwiemy ją "BootX.BBLib" to będzie ona wczytywana automatycznie, jeśli zaś nie, to możemy ją załadować do BootX'a komendą *Load bootblock library* z menu *Project*. Proste?

I jeszcze kilka słów dla osób, które zainteresował niecodzienny wygląd obu listingów. Wykorzystują one szerokie, a zarazem niezbyt znane możliwości ASM-One'a w zakresie makrodefinicji i dyrektyw asemblacji, warto je więc choćby skrótowo omówić.

Dyrektywa `set` odpowiada działaniem `equ`, z tym, że tworzona jest etykieta, której wartość można w toku asemblacji modyfikować. `Boot macro` oznacza początek makrodefinicji o nazwie `Boot`. Pojawiające się za etykietą `name` backslah i "małpa" (\@) zapobiegają wystąpieniu błędu `Double label`, który zostałby zasygnalizowany w przypadku więcej niż jednego odwołania do makrodefinicji. `\1` i `\2` to parametry do niej przekazane; `fail` spowoduje wymuszenie błędu asemblacji, w tym kontekście wówczas gdy identyfikator przekroczy trzydzieści znaków długości (sprawdźcie!). Linia z `blk.b` ma za zadanie zadbać o uzupełnienie identyfikatora do owych trzydziestu znaków w przypadku gdy jest krótszy. Instrukcja pomiędzy `if1` a `endc` jest wykonywana tylko podczas pierwszej fazy asemblacji (Pass 1) i zapewnia nam automatyczne policzenie bootbloków, których ilość jest następnie w zmiennej `number` przekazywana do części informacyjnej biblioteki. `endm` jak nietrudno się domyślić wskazuje zakończenie tekstu makrodefinicji.

Z kolei w listingu nr 2 występuje dyrektywa `AUTO`. Jako jej parametry możemy podać serię dowolnych komend ASM-One'a, które podczas asemblacji zostaną wykonane tak jakby wpisano je w trybie bezpośrednim. I to by było na tyle.

Mirosław "Thorgal" Smyk

Amos cz.II

W drugiej części opisu do Amosa omówimy zastosowanie procedur i etykiet, oraz zapoznamy się z wieloma nowymi, przydatnymi rozkazami. Na początku jednak warto będzie zająć się kompilatorem.

Kompilator (plik Compiler.AMOS) służy, jak sama nazwa wskazuje, do kompilowania programów, uprzednio napisanych za pomocą edytora i nagranych na dysk, ewenualnie na Ram dysk.

Program skompilowany (wynikowy), w odróżnieniu od programu źródłowego, jest plikiem wykonywalnym (ang. executable) i w pełni niezależnym od samego Amosa.

Obsługa kompilatora jest banalnie prosta: wystarczy wczytać plik Compiler.AMOS do edytora, uruchomić go (funkcja *Run*) i kliknąć na napisie *'COMPILE'*.

Otworzy się wtedy okienko do wybrania pliku, który chcemy skompilowć, a następnie pojawi się identyczne okienko umożliwiające wpisanie nazwy pliku wynikowego, lub wybranie pliku znajdującego się już na dysku (jeżeli chcemy, aby nasz program nagrał się bezpośrednio "na" nim).

Proces kompilacji trwa około 3 minut (dla krótkich programów), a jego stopień zaawansowania w pracy pokazuje efektownie wykonany "licznik".

Niestety poważną wadą skompilowanych plików jest ich długość - około 50 Kb, nawet gdy nasza "źródłówka" będzie zawierała zaledwie kilka instrukcji.

Nie znaczy to, że kilka instrukcji aż tyle zajmuje! Wspomniane 50 Kb stanowi bazę dla wszystkich napisanych w naszym programie rozkazów i długość programu wynikowego będzie mniej więcej równa 50 Kb + długość pliku źrodłowego.

Powyższy fakt praktycznie uniemożliwia zastosowanie Amosa do tworzenia króciutkich programów pomocniczych np. typu Memory Info (do podawania ilości dostępnej pamięci Ram).

Szybkość skompilowanego programu jest taka sama jak w edytorze (może troszeczkę szybsza).

Nie jest to wadą ze względu na to, iż programy z edytora wykonywane są już maksymalnie szybko, a ponadto, byłoby to niezbyt miłą niespodzianką, gdybyśmy napisali grę, która po skompilowaniu działałaby zupełnie inaczej niż uruchamiana z edytora.

Kompilator ma jeszcze kilka opcji, ale że powyższy opis wystarczy do udanej kompilacji, przechodzimy więc do procedur.

PROCEDURY

Często może się zdażyć, iż w kilku miejscach naszego programu będziemy potrzebowali wykonać pewne obliczenia. Nie ma sensu wielokrotne przepisywanie tych samych fragmentów, lecz znacznie wygodniej jest napisać algorytm obliczeń jeden raz, a później tylko się do niego odwoływać.

W takim przypadku pomocne stają się procedury. Zastosowamie procedur jest szerokie i nie tylko służą one do wykonywania obliczeń, ale równie dobrze można traktować je jako np. podprogramy.

```
Procedure "nazwa"  -deklaracja procedury "nazwa"
komendy            -ciąg komend
End Proc           -koniec procedury
```

Tak zadeklarowaną procedurę możemy wywoływać w dowolnym miejscu naszego programu, poprzez napisanie jej nazwy. Należy jednakże pamiętać o pewnych zasadach:

1. Zmienne i tablice używane wewnątrz procedur nie mają nic wspólnego ze zmiennymi z reszty programu, mimo że mogą posiadać identyczne nazwy. Aby móc korzystać z wartości zmiennych lub tablic, używamy komendy `Shared`. np.:

```
A$="Amiga"
KEBAB
Procedure KEBAB
Shared A$ - pozwala na użycie zmiennej A$ w
Print A$                  procedurze KEBAB.
End Proc
```

Gdy napisaliśmy wiele procedur korzystających ze zmiennych używanych w programie głównym, to wygodniej będze posłużyć się rozkazem `Global`. np.:
`Global A$,X,TXT$(),T()` -powoduje, że zmienne A$ i X, oraz tablice TXT$() i T() stają się zmiennymi /tablicami globalnymi, tj. rozpoznawanymi w każdym miejscu programu (o tablicach będziemy mówić za chwilę).

2. Wyjście z procedury następuje po wykonaniu wszystkich zawartych w niej instrukcji, lecz użycie komendy `Pop Proc` spowoduje wcześniejsze zakończenie jej działania i powrót (podobnie jak w przypadku `End Proc`) do następnej linii po miejscu z którego została wywołana.

3. Innym sposobem na korzystanie ze zmiennych z głównego programu, a następnie wyprowadzenie wyniku "na zewnątrz", jest użycie parametrów (nazw zmiennych) przy nazwie procedury i przy `End Proc`, jednakże komenda `Global` jest wygodniejsza i bardziej według mnie elegancka.

Wartość zmiennej przy `End Proc` zos- taje przypisana specjalnej zmiennej, odpowiednio: `Param`, `Param$` lub też `Param#` -tej ostatniej w przypadku użycia zmiennej rzeczywistej. W tym miejscu należy powiedzieć, że zmiennych rzeczywistych (np. A#, S1#) używamy wtedy gdy przewidujemy ułamkową postać zmiennej, na przykład wtedy kiedy ją dzielimy, wyciągamy z niej pierwiastek, itp.

4. Procedurę można wywołać nie tylko przez napisanie jej nazwy, ale i również przez wpisanie `Proc nazwa`. Jest to bez różnicy w działaniu programu, jedynie zwiększa jego przejrzystość w edytorze.

5. Cały zapis procedury możemy "zwinąć" za pomocą dostępnej z edytora funkcji *Fold/Unfold*, również w celu zwiększenia przejrzystości zapisu.

Inną pomocną rzeczą jest stosowanie etykiet.

W dowolnym miejscu naszego programu wystarczy napisać jej nazwę zakończoną dwukropkiem... i cała formalność z głowy! Teraz gdy zechcemy aby program "skoczył do etykiety", tj. zaczął wykonywać instrukcje napisane po deklaracji, wystarczy wpisać `Goto nazwa_etykiety` (ale bez dwukropka). np.:

```
A=5
ETYK:
Print "AMOS"
A=A-1
If A<0 Then Goto ETYK
If A=0 Then Print "Wystarczy"
```

W pisanym przez nas programie mogą wystąpić błędy. Do ich obsługi stosujemy instrukję `On Error`. Poniższe jej zastosowanie nie odnosi się do błędów składniowych (wykryłby je już sam edytor), lecz do błędów semantycznych, czyli wynikłych z powodów, na które programista często nie ma wpływu, takich jak na przykład błędne wprowadzenie danych przez użytkownika, wyjęcie dysku przed rozpoczęciem nagrywania itd. Deklaracje obsługi błędu:

`On Error Goto ETYKIETA`
-w razie błędu skok do etykiety.

`On Error Proc PROCEDURA`
-w razie błędu skok do procedury.

Dzięki tym deklaracjom, w przypadku błędu, Amos zacznie wykonywać odpowiednią procedurę lub instrukcje następujące po etykiecie. Ciąg komend musi kończyć się na instrukcji `Resume`, np.

`Resume` - powraca do komendy w której wystąpił błąd.

`Resume Next` - przechodzi do następnej instrukcji znajdującej się po instrukcji wywołującej błąd.

`Resume ETYKIETA` - przechodzi do etykiety.

Bezpośrednio po deklaracji `On Error`, możemy wpisać `Resume Label ETYKIETA`, a przejście do

etykiety wywołać później przez `Resume Label`. Pomocnymi funkcjami przy testowaniu programów, są:

`Error N` - wywołuje błąd o numerze N.

`Errn` - przyjmuje wartość numeru ostatniego błędu.

TABLICE

Tablice deklarujemy rozkazem `Dim`. Ale moment... Co to są właściwie tablice? Otóż są one w pewnym sensie magazynami, w których przechowujemy dane. Zamiast tworzyć wiele zmiennych, wygodniej będzie zadeklarować tablicę i w niej umieszczać dane, co ułatwi do nich dostęp i podniesie przejrzystość programu. Aby możliwe było użycie tablic, należy je najpierw zadeklarować, czyli zarezerwować dla nich pamięć:

`Dim A(9)` - deklaruje tablicę A() z możliwością "zmagazynowania" dziesięciu liczb (pierwsza dana ma numer zero).

`Dim A$(9)` - jak wyżej, ale przechowywać będziemy tekst.

Dane do tablic wprowadzamy w następujący sposób:

`A(0)=3 : A(1)=100 : A(2)=0 : A$(8)="Kebab"`.... itd

Tablice mogą być wielowymiarowe, tzn. że położenie danej określać może więcej liczb. Dzieje się to przykładowo na zasadzie układu współrzędnych : dana ma swoje współrzędne X,Y,Z - tablica trójwymiarowa. Poniższy przykład ilustruje zastosowanie tablicy dwuwymiarowej. Oczywiście zamiast nazwania tablicy "A", możemy wymyślać dowolne, dłuższe nazwy np. TABLICA$(10).

```
Dim A(2,2)
A(0,0)=1 : A(0,1)=2 : A(0,2)=3 : A(1,0)=4 : A(1,1)=5
A(1,2)=6 : A(2,0)=7 : A(2,1)=8 : A(2,2)=9
For F=0 To 2
For A=0 To 2
Print A(F,A)
Next
A=A+1
Next
```

Nie jest to może zastosowanie praktyczne, nie mniej jednak obrazuje omawiany problem. Pozotałe instrukcje do pracy z tablicami:

`Sort A(0)` - sortuje dane w tablicy od wartości najmniejszej do największej.

`Match (A(0),n)` - szuka w tablicy danej n (lub tekstu w wypadku tablicy tekstowej), oraz przybiera wartość określającą jej położenie. W przypadku nie znalezienia takowej, funkcja ta przyjmie ujemną wartość określającą położenie danej najbardziej zbliżonej wartością do poszukiwanej. Komendy `Match` używamy jedynie do jednowymiarowych i posortowanych tablic.

A teraz pora na obiecaną listę niektórych instrukcji wraz ich z wyjaśnieniem.

Bell x	odgłos dzwonka o częstotliwości x
Boom	odgłos wybuchu
Break On	włącza możliwość przerwania programu klawiszami Ctrl-c
Break Off	wyłącza ją
Close Workbench	zamyka ekran Workbencha, dzięki czemu rośnie ilość wolnej pamięci
Close Editor	jak wyżej, ale w stosunku do edytora.
Dec x	zmniejsza o jeden wartość zmiennej x (dekrementacja)
Direct	zatrzymanie programu i przejście do trybu bezpośredniego
Edit	zatrzymanie programu i przejście do edytora
End	tylko zakończenie programu
Inc X	zwiększa o jeden wartość zmiennej X (inkrementacja)
Led Off	wyłącza filtry (ściemniona dioda Power)
Led On	włącza filtry, włącza diodę
Llist	drukuje listing programu
Lprint "tekst"	drukuje podany "tekst" lub zmienne
Rem	służy do umieszczania komentarza
Set Buffer x	ustala rozmiar w x kilobajtach obszaru zmiennych. Instrukcja ta może być użyta wyłącznie na początku programu
Shoot	odgłos strzału
Wait Key	czeka na wciśnięcie klawisza
Wait x	czeka 1/50sek*x

Podane instrukcje nie mieszczą się w zagadnieniach, które będziemy omawiać w kolejnych numerach, dlatego też zapoznaliśmy się z nimi już teraz. Reszta materiału przedstawiona zostanie w formie działów np. Grafika, Muzyka, Tekst itd. Do zobaczenia za miesiąc!

Zbigniew "Zybul" Piotrowicz

High Speed Pascal 1.0

po raz drugi - okiem programisty

W numerze 5-tym 'Kebaba' zamieszczony został opis nowego kompilatora jezyka Pascal na Amigę - High Speed Pascala 1.0. Podano wtedy (zgodnie z sugestią producenta), że jest to program zgodny pod względem języka z bardzo znaną implementacją Pascala na IBM PC - Turbo Pascalem 5.0. Od czasu ukazania się tamtego artykułu miałem sposobność zetknąć się bliżej z HS Pascalem szczególnie pod kątem zgodności z PC-towskim odpowiednikiem.

Firma *HiSoft* (m.in. twórca znanego *Devpac'a)* włożyła w swe dzieło dużo pracy, nie ustrzegła się jednak od kilku niezgodności a nawet błędów. Miałem okazję przeprowadzić kilka razy próbę przeniesienia dość uniwersalnego kodu (bez trików sprzętowych itp.) z PC na Amigę. Pierwsze wrażenia były pozytywne, lecz przy bliższej analizie udało mi się ujawnić kilka niezgodności utrudniających swobodne przenoszenie kodu źródłowego miedzy PC a Amigą.

Sadzę, że znajomość tych ograniczeń oszczędzi Czytelnikom tego artykułu sporo czasu lub nawet ewentualnego zawodu.

Wszystkie problemy podzieliłem na cztery grupy:

1) różnice syntaktyczne - wykrywane na poziomie kodu źródłowego, wymagające przeróbki programu.

2) różnice polegające na odmiennym zachowaniu tego samego kodu na Amidze i PC.

3) różnice nie wykrywane w fazie kompilacji wywołujące natomiast błędy wykonania.

4) specyficzne cechy środowiska HS Pascala mogące powodować trudności użytkownikowi.

A oto spis zauważonych przeze mnie niezgodności i błędów:

Grupa 1.

a) Funkcje `ParamStr` i `ParamCount` zostały umieszczone w module DOS i wymagają użycia komendy `"Uses DOS"` w części deklaracyjnej programu. (Standardowo znajdują się one w module SYSTEM i są dostępne bezpośrednio)

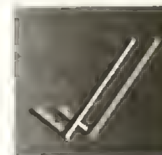
b) Użycie komendy `"uses"` w sekcji `"implementation"` wywołuje błąd - uniemożliwia to pisanie modułów wzajemnie się wywołujących. (W TP 5.0 jest to możliwe). Nie jest to co prawda objawem zbyt pięknego stylu programistycznego lecz niektórzy piszą takie moduły bądź umieszczają `"uses"` w sekcji `"implementation"` zamiast w `"interface"`.

c) Dyrektywy kompilatora zawierające ścieżki do pliku np. $I (include file) pisane zgodnie z konwencją MS-DOS nie są poprawnie interpretowane i wymagają poprawek. (to dosyć prosta do poprawienia niezgodność). Przykładowo plik c:\files\a.pas może przyjąć postać df0:files/a.pas

d) Funkcje typu `"GetFAttr"` czy `"SetFAttr"` używają w każdym z systemów własnych flag dla plików. Wymaga to każdorazowo poprawki.

Grupa 2.

a) Dużo zawodu przyniosło mi wynalezienie kilku "atrap" w module GRAPH. Kilka funkcji nie zostało po prostu zaimplementowanych choć są one rozpoznawane przez kompilator (nie wywołują jednak przypisanych im efektów). Dotyczy to w szczególności funkcji związanych z wyświetlaniem tekstów w trybie graficznym. Przykładowo niemożliwa jest zmiana czcionki, rozmiaru i kierunku wyświetlania tekstu `(SetTextStyle)`.



b) Zarówno w trybie tekstowym jak i graficznym używany jest typowy amigowski zestaw znaków. Oznacza to, że wszelkie programy wykorzystujące semigrafikę nie będą zbyt pięknie wyglądały. Na szczęście prostym i szybkim rozwiązaniem jest modyfikacja systemowej czcionki (postaram się umieścić taką na PD Pack'u).





c) Podobny problem istnieje z odczytem klawiatury za pomocą funkcji `"ReadKey"`. Kody klawiszy specjalnych takich jak kursory, klawisze funkcyjne itp. są inne. Rozwiązaniem byłoby opracowanie nowego SetMap'u (też w przyszłości na PD-Pack'u).

d) Dość poważny problem wiąże się z sposobem implementacji trybu tekstowego na Amidze. Z faktu, że jest to zwykła konsola wynika wiele cech nie występujących w TP 5.0:

- można zatrzymać wydruk (np. write) wciskając dowolny klawisz alfanumeryczny.

- tryb highvideo daje efekt w postaci tekstu pogrubionego.

Grupa 3.

a) Udostępniony driver SVGA `(InitGraph)` ma tę wadę, że nie

pracuje w żadnym z 19-tu dostępnych trybów graficznych. Próba uruchomienia grafiki z tym sterownikiem kończy się zawsze błędem wykonania.

b) Próba uruchomienia sterownika nr 19 kończy się GURU! - podobnie jak z niektórymi zbyt dużymi wartościami trybów dla innych sterowników. (W zasadzie nie jest to błąd bo formalnie ten sterownik i te tryby nie istnieją. Nie zdarzy to się przy użyciu właściwego sterownika we właściwym mu trybie. Miłośnicy pętli for mogą jednak być niezadowoleni - to co na PC po prostu nie działa, na Amidze załamuje cały system!)

c) Podobnie jak w punkcie 1c należy zwrócić uwagę na nazwy zbiorów zawarte w zmiennych stringowych. Będą wymagały modyfikacji wszelkie funkcje przetwarzające ścieżki do zbiorów (ze względu na różnice notacji w Amiga DOS i MS-DOS). Dotyczy to wszelkich procedur komunikujących się z dyskowym systemem operacyjnym (`assign`, `chdir`, `mkdir`, `rmdir`, `findfirst` etc.).

Grupa 4.

a) Przypadkowo udało mi się wykryć błąd w środowisku HS Pascal'a. Aby samemu to sprawdzić załaduj program, wciśnij CTRL-C (kompilacja) - aby załadował się zbiór Pascal.lib (odpowiednik turbo.tpl), wcisnij klawisz C (Continue) i Enter - niezawodna metoda na wywołanie GURU.

b) Dość uciążliwy jest inny błąd. W menu Pascal Config (CTRL-O) należy podać katalogi w okienkach "`Output dir`" czy "`Units`". Jeśli jednak podamy jako katalog samo urządzenie logiczne (np. RAM: albo DF0:) to kompilator niewłaściwie "sklei" jego nazwę z nazwą zbioru (Np. RAM: i file.pas otrzymamy RAM:/file.pas) w wyniku czego nastąpi błąd. Urządzenie logiczne nie może zatem być katalogiem wyjściowym lub zawierającym moduły. Należy odpowiednio wcześniej założyć jakikolwiek katalog na urządzeniu logicznym i odpowiednio zmienić Pascal Config (np. Units: RAM:output/).

Spis ten nie jest oczywiście pełny. Jeśli któryś z Czytelników miał inne problemy z HSP to proszę o listy. Postaram się zebrać wszelkie informacje na ten temat i ponownie opublikować.

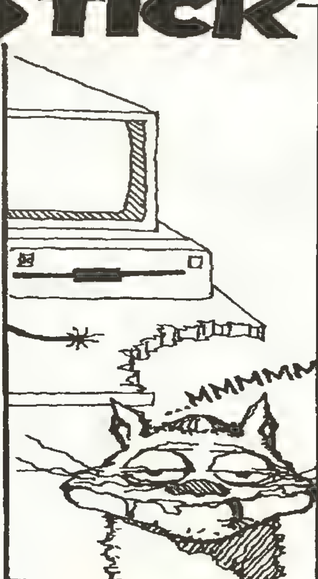
Na koniec chciałbym przekazać pewnie już pesymistycznie nastawionym Czytelnikom parę ciekawych informacji o tym co DZIAŁA w HS Pascalu całkiem dobrze.

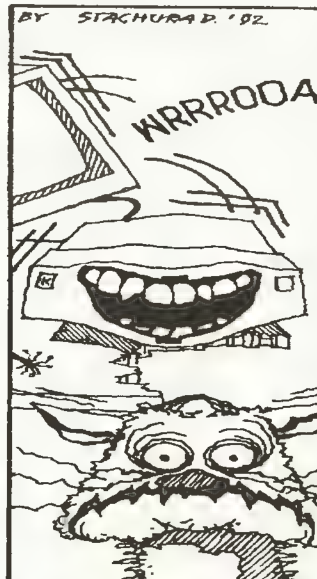
W poniższej tabeli znajdziecie wszystkie dostępne sterowniki graficzne oraz odpowiednie tryby. Jak łatwo zauważyć Amiga potrafi emulować całkiem sporą ilość kart graficznych. Dla tych którzy nie wiedzą jak użyć instrukcji "`initgraph`" polecam podręczniki Turbo Pascala autorstwa pana Marciniaka (ew. Bieleckiego, a fe!).

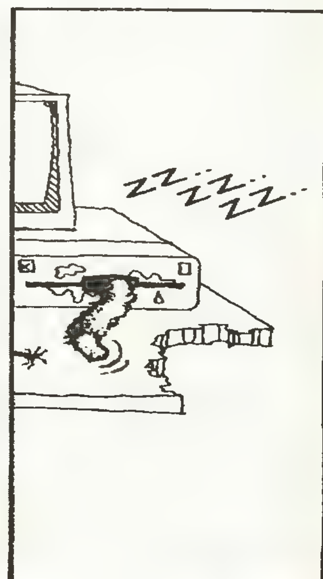
Dostępne sterowniki graficzne:

nr	nazwa sterownika:
0	DEFAULT
1	CGA
2	MCGA
3	EGA
4	EGA64
5	EGAMONO
6	IBM8514
7	HERCMONO
8	ATT400
9	VGA
10	SVGA
11	PC3270
12	STCOLOR
13	STMONO
14	AMIGA









Dostępne tryby i ich rozdzielczości					
nr	rozdzielczość	nazwy trybów	nr	rozdzielczość	nazwy trybów
0	320/200/4	CGAC0,MCGAC0,ATT400C0	10	640/480/16	VGAHI,IBM8541LO,IBM8541HI
1	320/200/4	CGAC1,MCGAC1,ATT400C1	11	640/256/2	PC3270HI
2	320/200/4	CGAC2,MCGAC2,ATT400C2	12	320/200/16	STLOW
3	320/200/4	CGAC3,MCGAC3,ATT400C3	13	640/200/4	STMEDIUM
4	640/200/2	CGAHI,MCGAMED,ATT400MED	14	640/400/2	STHIGH
5	640/480/2	MCGAHI,ATT400HI	15	640/256/4	AMIGAWB
6	640/200/16	EGALO,EGA64LO,VGALO	16	640/256/16	AMIGAWB16
7	640/350/8	EGAHI,VGAMED	17	320/256/32	AMIGALO32
8	640/350/4	EGA64HI	18	640/265/4	AMIGACUSTOM
9	640/350/2	EGAMONOHI,HERCMONOHI			

Oczywiście cały czas czekam na uwagi i zapytania. Chętnie umieścimy na łamach Kebaba rubrykę mającą na celu wymianę doświadczeń początkujących i zaawansowanych programistów. Piszcie!

K. Moroń

Mapa Pamięci Amigi (c.d.)

Witam Was w kolejnym, nieco innym odcinku cyklu. Nieco innym, gdyż dostałem ostatnio dokumentację do nowego Blittera (ECS) i chciałbym teraz podzielić się z Czytelnikami informacjami z tamtąd płynącymi, a za miesiąc wracamy ponownie do rozpoczętych rejestrów.

Jakie korzyści płyną z nowego ECS'a? Najprościej przedstawić je w następujących punktach:

1. Większy obszar adresowania. Nowy ECS posiada możliwość adresowania 2MB pamięci, zatem do takiej wartości możemy obecnie rozszerzyć naszą pamięć CHIP.

2. Większe zakresy danych. Dzięki temu możemy operować na znacznie większych oknach danych (32768 pixeli na 32768 linii), oraz kreślić znacznie dłuższe linie.

3. Nowe tryby graficzne. W trybie SuperHires mamy możliwość wyświetlania obrazu o rozdzielczości 1280x512x4, w trybie Productivity 640x960x4, oraz w trybie A2024 1008x1024 w 4 odcieniach szarości.

4. Poszerzone możliwości współpracy z Genlockiem.

A oto opis wybranych rejestrów:

05C BLTSIZV W A

Pionowa wielkość okna danych dla Blitera. Piętnaście bitów (0-14) tegoż rejestru definiuje pionową wielkość okna na którym będziemy wykonywać operacje. Jak łatwo obliczyć 2 do potęgi 15 daje wartość 32768, co jest naszą maksymalną wysokością okna.

05E BLTSIZH W A

Pozioma wielkość okna danych i start. Bity 0-10 definiują szerokość okna wyrażoną w słowach. Należy pamiętać, aby rejestr ten zapisywać na końcu, gdyż umieszczenie w nim jakiejkolwiek wartości powoduje rozpoczęcie pracy Blittera. Przy okazji powyższych dwóch rejestrów nasuwa się pytanie, co stało się z rejestrem BLTSIZE (058). Otóż nic się nie stało. Dla zachowania kompatybilności z poprzednimi programami zostawiono go w niezmienionym stanie. Dlatego też, Blitera możemy uruchamiać obecnie w dwojaki sposób. Nie wolno jednak zapominać, że operacje na dużych oknach można wykonywać jedynie przy pomocy powyższej pary rejestrów.

020 DSKPTH W A

Wskaźnik danych dla dysku. W rejestrze tym dodano 2 bity aby umożliwić dostęp do pełnych 2 megabajtów pamięci CHIP.

050 BLTxPTH W A

Wskaźnik kanału x dla Blitera. Rozszerzone o dwa bity w celu umożliwienia dostępu Blitera do 2 megabajtów pamięci.

080 COP1LCH W A
084 COP2LCH W A

Wskaźnik Coperlistu. Rozszerzone o dwa bity.

0A0 AUDxLCH W A

Adres danych dla x kanału audio. Rozszerzone o dwa bity.

02E COPCON W A

Rejestr kontroli koprocesora graficznego. W standardowym Blitterze ustawienie pierwszego bitu tegoż rejestru (CDANG) umożliwiało Copperowi zapis rejestrów Blitera. Obecnie w ECS'ie, gdy bit ten zostaje ustawiony, automatycznie oznacza to możliwość zapisu wszystkich rejestrów Amigi.

1E4 DIWHIGH W A D

Rejestr definiujący wielkość wyświetlanego okna (border). Amiga posiada dwa rejestry DIWSTOP oraz DIWSTRT. Sterujące wielkością wyświetlanej ramki na ekranie. Rejestr DIWHIGH jest rozszerzeniem wspomnianych rejestrów i tak poszczególne bity oznaczają:

bity 15-14
Nie używane, skasować.

bit 13
Najstarszy bit poziomego "stop".

bity 12-11
Nie używane, skasować.

bity 10-8
Najstarsze bity pionowego "stop".

bity 7-6
Nie używane, skasować.

bit 5
Najstarszy bit poziomego "start".

bity 4-3
Nie używane, skasować.

bit. 2-0
Najstarszy bit pionowego "start".

004 VPOSR R A

Odczyt najbardziej znaczących bitów pozycji pionowej rastra. Poszczególne bity oznaczają:

bit 15
(LOF) Używany w trybie Inter lace.

bity 14-7
Oznaczają rodzaj Agnusa.
bity 6-3
Nie używane.

bity 2-0
Najstarsze bity pozycji pionowej rastra.

Jak widać bity 14-7 oznaczają rodzaj aktualnie podłączonego Agnusa. W przypadku podłączenia ECS'a bit 13 jest ustawiony, jednak ta informacja nie wystarcza aby stwierdzić, że na pewno w testowanej Amidze znajduje się ECS. Patrz rejestr następny.

07C DENISEID R D

Numer identyfikacyjny układu Denise. Bity 15-8 są zarezerwowane. Natomiast jeśli bity 7-0 zawierają wartość $FC to przy ustawionym bicie 13 z rejestru VPOSR (004) możemy być pewni, że jesteśmy szczęśliwymi posiadaczami rozszerzonego ECS'a.

100 BPLCON0 W A D

Rejestr kontroli wyświetlanego obrazu. Bit 6 (SHRES) tegoż rejestru włącza tryb SuperHires. Ilość BitPlanów (1 lub 2) wpisujemy w miejsce bitów 14-12.

Krzysztof Kobus

Reset? Czemu nie...

Z pewnością nie raz mieliście okazję oglądać programy, głównie różnego rodzaju demonstracje, w których po naciśnięciu klawiszy *Ctrl-Amiga-Amiga* zamiast procedury resetującej komputer uruchamiała się np. następna część dema. Pytasz jak to możliwe? Otóż programy te wykorzystują fakt istnienia w Amidze wektorów systemowych odpowiedzialnych za reset a także mechanizmu tzw. "ciepłego" startu (ang. *warm start*), który uruchamiany jest właśnie poprzez wciśnięcie wyżej wymienionych klawiszy.

W odróżnieniu do "zimnego" *(cold) startu*, mającego miejsce np. po włączeniu zasilania, podczas *warm* startu nie jest kasowana pamięć RAM komputera, więc programy lub dane które w niej się znajdowały, pozostają nienaruszone. Fakt ten wykorzystują między innymi autorzy różnego rodzaju ripper'ów.

Rozpocznijmy zatem od poznania niezbędnych, wykorzystywanych w tym celu, zmiennych systemowych. Interesują nas głównie wektory *ColdCapture* i *CoolCapture* oraz zmienna *ChkSum*.

Do czego one służą? Otóż *ColdCapture* zajmuje cztery bajty (długie słowo), i zaczyna się od 42

bajtu względem bazy Exec.library (przypomnijmy: aby otrzymać adres *ColdCapture* należy do bazy Exec'a znajdującej się pod adresem absolutnym $0004 dodać przesunięcie, czyli właśnie 42). Jeśli wartość wektora jest równa zero, to podczas inicializacji komputer zachowuje się "normalnie", jeśli zaś wartość jest różna od zera zostaje potraktowana jako adres i komputer wykona skok do umieszczonego pod tym adresem programu.

Podobnie ma się sprawa z *CoolCapture* (46 bajtów od bazy). Różnica między tymi wektorami polega na momencie testowania przez komputer ich zawartości. *ColdCapture* jest sprawdzany we wczesnej fazie inicjalizacji Amigi, gdy nie ma jeszcze w pamięci struktur systemowych, zaś *CoolCapture* w fazie końcowej tego procesu, już po otwarciu niezbędnych bibliotek.

Jak się już zapewne domyśliliście wpisanie do jednej z powyższych komórek adresu naszej własnej procedury powinno spowodować jej wykonanie przez procesor, przy okazji najbliższego resetu (chyba, że jakiś inny program zmieni te wektory do własnych celów).

Pozostała nam jeszcze ostatnia, równie ważna, zmienna o nazwie *ChkSum*. Zawiera ona sumę kontolną (*CheckSum*) fragmentu struktury Exec, zwanego *Static System Variables*. Co to takiego? *Static System Variables* to grupa 13 zmiennych systemowych (rozpoczynająca się od 34 bajtu struktury Exec) w której skład wchodzą także wektory *ColdCapture* oraz *CoolCapture*. Jak każda ważniejsza struktura systemowa, także ta posiada swoją sumę kontrolną, którą należy aktualizować PO wszelkich modyfikacjach zawartych w niej zmiennych. W przypadku wystąpienia błędnej sumy kontrolnej, system zignoruje wartości zmiennych zawartych w *Static System Variables*.

Obliczenie owej sumy kontrolnej dla tego obszaru polega polega na dodaniu do siebie kolejnych 46 bajtów pamięci (tyle bowiem miejsca zajmują owe zmienne) a następnie zanegowaniu tak otrzymanej sumy (np.instrukcją NOT). Rezultat tej operacji należy niezwłocznie wstawić do zmiennej *ChkSum* pamiętając, że jest ona 2 bajtowa (adres: 82 bajty od bazy Exec).

I gotowe...

Na koniec parę uwag:

... dla użytkowników Amigi 2000 (B i C):

- ten model komputera, w odróżnieniu od Amigi 500 posiada zmieniony mechanizm "ciepłego" startu. W przypadku resetowania, różnica polega na ... dokumentnym zerowaniu pamięci RAM, co niestety eliminuje możliwość uruchamiania jakichkolwiek procedur w wyżej wymieniony sposób. *

... dla wszystkich:

- jeżeli chcesz aby twoja procedurka (korzystająca z ColdCapture) była aktywowana po każdym resecie, to musisz ustawiać ten wektor po każdym jej wywołaniu (jest on bowiem zerowany). W przypadku *CoolCapture* nie jest to konieczne.

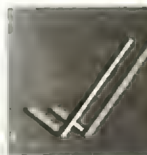
- jeśli chcesz jednocześnie wykorzystać oba wektory np. umieszczając dwa różne programy, to musisz wiedzieć, że w przypadku resetu wykonany zostanie tylko jeden z nich (używający *ColdCapture*). Powtórne zresetowanie nie wywoła (jak mogłoby się wydawać) procedury podwieszonej pod *CoolCapture*.

- powyższy efekt można uzyskać jedynie przy aktywnym systemie. Jeśli wskaźnik bazy exec.library (bądź sama biblioteka) zostanie zamazany to reset spowoduje pełną inicjację systemu (wraz z zerowaniem pamięci RAM).

- niektóre z dostępnych na rynku przystawek (np. Action Replay) mają wbudowane mechanizmy uniemożliwiające uruchomienie tak "podczepionych" programów.

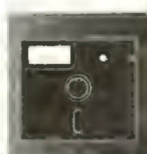
Z jednej strony zabezpiecza to użytkownika przed niecnym atakiem rozmaitych podstępnych wirusów, ale drugiej zaś, uniemożliwia przetestowanie naszej procedurki. Jedynym na to lekarstwem jest... odłączenie owego urządzenia na czas testowania procedury.

Marcin 'MR SOFT' Orłowski



* Spowodowane to jest drobnym błędem w systemie operacyjnym (do V1.3), którego twórcy nie przewidzieli możliwości zainstalowania w Amidze więcej niż 512KB Chip-RAM'u.

W przypadku zainstalowania (również w A500) 1MB w/w pamięci, następuje reset z całkowitym kasowaniem zawartości pamięci. Uniemożliwia to m.in. poprawne korzystanie z "resetoodpornego" RAM-dysku "RAD:". Aby temu zaradzić należy w "startup-sequence" (lub później) podać komendę "setpatch r", która odpowiednio modyfikuje właściwy fragment sytemu. (przyp. red.)



UWAGA CZYTELNICY

Aby umożliwić Wam kupno "KEBAB"a poza oficjalną dystrybucją, uruchomiliśmy kolejne punkty sprzedaży (na razie na terenie Szczecina):

- sklep **"ADMIRAL - COMP"** ul. Monte Casino 37
- sklep "PROFIX" ul. Ściegiennego 4
- stoisko "SONIX" IIp. PDT "Posejdon"
- sklep "MIMAX" ul. J. Piłsudskiego 34
- sklep "AMISOFT" ul. Niepodległości 38A
- stoisko "ATACOM" Ip. "MULTI" Os. Przyjaźni

Jak scrunchować demo?

Wydawać by się mogło, że po napisaniu choćby prostego dema, a co za tym idzie, poznaniu obsługi specjalizowanych układów Amigi, nic nie jest już w stanie zaskoczyć początkującego kodera. Niestety, do dziś poważnym problemem pozostał sposób takiego przekształcenia kodu napisanego na assemblerze (np. na MasterSece), aby nasze demo można było uruchomić z poziomu CLI lub WorkBencha. Artykuł ten ma na celu rozwianie podobnych wątpliwości i ostateczne rozwiązanie problemu.

Załóżmy, że nasze demo będzie składać się z głównego kodu, modułu muzycznego, logosa, oraz fontów. Zarówno kod dema, jak i poszczególne jego dane muszą posiadać własną lokację (swój adres) w pamięci. Przyjmijmy zatem następujący, przykładowy podział pamięci dla poszczególnych składowych naszego dema:

moduł muzyczny:
$30000 - $4A348

kod główny:
$50000 - $5689A

logos:
$58000 - $60000

fonty:
$61000 - $66000

W ten sposób dokonaliśmy podziału pamięci. Jak jednak spowodować, aby poszczególne elementy znalazły się na swoich miejscach? W przypadku danych należy na początku tekstu źródłowego umieścić linie ładujące z dyskietki (twardego dysku) powyższe pliki do pamięci oczywiście pod odpowiednie adresy. Służy do tego dyrektywa >extern assemblera MasterSeka. Wpisujemy zatem trzy linie:

```
>extern"df0:mod.NazMod",$30000

>extern "df0:Logos",$58000

>extern "df0:Fonty",$61000
```

Teraz, po zasemblowaniu kodu źródłowego komendą "A", i wykonaniu komendy"Y", wyżej wymienione pliki danych zostałyby wczytane pod odpowiednie adresy.

Zanim jednak to uczynimy na początku kodu źródłowego wpiszemy jeszcze dwie linie:

```
org $50000

load $50000
```

które spowodują, że nasz kod główny również zostanie umieszczony we właściwym miejscu. W tej chwili możemy juz spokojnie zasemblować kod, następnie (komendą Y) wczytać wspomniane pliki i całość tak przygotowanej pamięci zapisać na dyskietce pod nazwą na przykład "Demo1".

Wykonujemy zatem komendę "WI" i jako parametry podajemy adresy początku: $30000, oraz końca: $66000. W ten właśnie sposób przygotowaliśmy demo do crunchingu, którego można dokonać przy pomocy dwóch programów: *ByteKillera* oraz *DefjamPackera V3.2*. Tutaj ograniczę się do opisu tego drugiego tym bardziej, że ByteKillera obsługuje się znacznie prościej.

Uruchamiamy zatem DefjamPackera i jesteśmy pytani czy chcemy pracować w trybie MegaCrunch - wybieramy odpowiedź N.

Następnie jesteśmy proszeni o podanie dolnej i górnej granicy zajmowanej pamięci. W naszym przypadku chodzi o zakres $30000-$66000, niemniej jednak zawsze warto zostawić pewien margines bezpieczeństwa, zatem podajemy np. $28000-$70000.

Program alokuje teraz odpowiedni obszar pamięci, równocześnie czyszcząc go i pyta o efektywność crunchowania.

Wszyscy pracujący choć trochę z komputerami wiedzą, iż zazwyczaj lepszą wydajność uzyskujemy kosztem czasu.

Aby zatem nie zostawiać komputera kompresującego nasze dane przez 46 godzin, na pytanie o "Scan Width" podajemy wartość $200, która to pozwoli na stosunkowo korzystny rezultat, w rozsądnym czasie. Teraz po wybraniu odpowiedzi "o" oznaczającej zwykły, nie relokowalny program ładujemy nasze demo "Demo1" pod adres w którym znajdowało się na początku czyli $30000.

Po załadowaniu stwierdzamy, że wszystkie pliki mamy wczytane, naciskamy RETURN, po którym następuje proces kompresowania. W zależności od tego, jak długie jest nasze demo udajemy się do kina bądź na kawę aby po powrocie podać programowi adres pod który ma skoczyć, w celu uruchomienia naszego dema. Zazwyczaj jest to początek kodu, zatem podajemy adres $50000.

Nie pozostaje już nic innego jak wybrać rejestr błyskania na ekranie podczas rozpakowywania dema, na następne pytanie o Pro-Decruncher dać odpowiedź N, zapisać rezultat na dyskietce pod nazwą Demo2 i pojechać z nim na najbliższe Copy-Party.

Krzysztof Kobus

P.S. Zarówno Defjam Packer jak i Byte Killer znajdują się na naszym pierwszym KEBAB PD disk'u

NTSC dla każdego

Przeglądając swego czasu jeden z numerów "C&A" natknąłem się na schemat usprawnienia sprzętowego dającego możliwość przełączania Amigi pomiędzy systemami PAL i NTSC. Przeróbka może i interesująca, bo rzeczywiście sporo gier i programów użytkowych pisanych dla ekranu o zmniejszonej wysokości charakterystycznego dla komputerów zza oceanu, prezentuje się lepiej w trybie NTSC. Szczególnie dużo zyskują gry, gdyż nabierają wyglądu podobnego do "coin-ops" - animacja jest bardziej płynna (60Hz), a ekran większy. Słowem - nic, tylko instalować...

Nim jednak sięgniesz Drogi Czytelniku po lutownicę dowiedz się, że jeśli jesteś posiadaczem Amigi wyposażonej w Fatter lub Super Agnusa, to identyczny efekt możesz osiągnąć bez rozbierania komputera i narażania się na utratę gwarancji.

Cały zabieg sprowadza się do skasowania w komórce BEAMCON0 ($dff1dc) bitu o numerze 5. Rejestr jest sprawdzany przez cały czas, zmiana będzie więc natychmiastowa. Minusem takiego rozwiązania są problemy jakie ma system operacyjny przy obsłudze otworzonych wcześniej ekranów przekraczających swą wysokością standardowe parametry NTSC. Przygotowałem więc program z **listingu 4,** który modyfikuje wektory restartu systemu i przełącza naszą Amigę na system NTSC tuż po wciśnięciu kombinacji Ctrl-Amiga-Amiga.

Następnie procedura samoczynnie dezaktywuje się, by na wypadek uszkodzenia jej przez uruchomiony program komputer nie zawiesił się podczas kolejnego "resetu". Wszystko co należy zrobić to "wklepać" na ASM-One'a lub Sekę listing 4, nagrać go na dyskietkę (przyda się nie raz), przeprowadzić asemblację opcją "a" i uruchomić przez "j", po czym "zresetować" maszynerię. A lutownicę schować na okazję montażu potencjometrów do regulacji palety barw Workbench'a.

Użytkownicy Kickstart'u 2.0, który samoczynnie przywraca właściwy tryb wyświetlania przy restarcie, muszą zadowolić się kasowaniem odpowiedniego bitu z poziomu monitora pamięci lub asemblera.

Sprawa ma się znacznie gorzej jeśli dysponujemy Amigą wyekwipowaną w stary model kości Agnus. Tu zmuszeni jesteśmy samodzielnie (procesorem bądź copperlistą) śledzić wiązkę elektronów omiatającą ekran pięćdziesiąt razy na sekundę i modyfikować jej położenie w odpowiednim momencie. Robi to **listing 3**, który czeka aż wspomniana wiązka osiągnie linię o numerze 260 ($104) i przenosi ją na koniec jej drogi po ekranie PAL, do linii 311 ($137), co w efekcie daje obraz skrócony o 51 (311-260) rastrów. Jako, że szybkość kreślenia wyno- si w przybliżeniu 15550 linii skaningowych na sekundę, częstotliwość tak otrzymanego obrazu osiąga około 60Hz (15550/260).

Metoda ta nie jest niestety tak uniwersalna jak poprzednia, niemniej można ją z niezłym skutkiem stosować we własnych programach, przy czym poprzez odpowiednią manipulację stałą "synchro" daje się podbić częstotliwość obrazu do ponad 62Hz, co widocznie poprawia jakość obrazu w trybie "interlace".

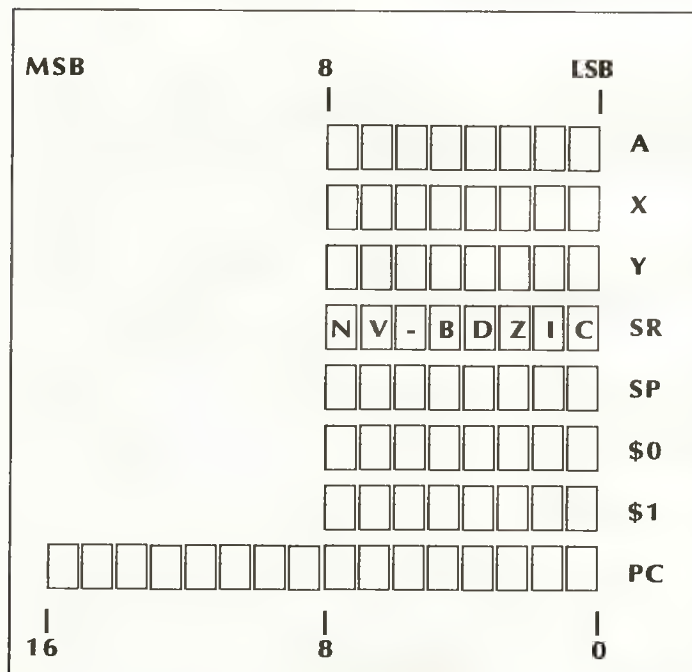
Miłosław "Thorgal" Smyk



Assembler

Po raz kolejny spotykamy się dzisiaj w gronie wytrwałych asemblerowców. Z naszej małej wycieczki w krainę VIC'a (w poprzednim odcinku), wrócimy na moment ponownie do procesora. Zanim podążymy dalej musimy jeszcze poznać kilka drobiazgów związanych z jego konstrukcją. Wspominaliśmy już wcześniej, gdy omawialiśmy rejestry procesora, że oprócz podstawowych rejestrów tj. Akumulatora oraz rejestrów indeksowych (X i Y) są jeszcze tzw. rejestry specjalnego przeznaczenia (patrz KEBAB 2-3/92). Z tych rejestrów specjalnych, nas zainteresuje dziś przede wszystkim rejestr "SR".

SR to skrót od angielskiego "Status Register". W polskiej literaturze spotyka się określenie "rejestr stanów procesora". Szczegółowy schemat wszystkich rejestrów przedstawia nam rysunek.



Widzimy, że rejestr "SR" ma wyszczególnione pojedyncze bity oraz przypisane im nazwy literowe. Rejestr ten zawiera siedem tzw. "flags" (patrz słowniczek). Czemu siedem, skoro bitów mamy osiem? Otóż po prostu jeden bit pozostał nie wykorzystany.

Rozpoczynając od lewej tj. od MSB mamy kolejno:

N - (N)egative Flag - znacznik ujemnego wyniku ostatniej operacji. Pojawia się (tzn. przyjmuje stan 1) gdy np. mamy wartość $00 a mimo to wykonamy zmniejszenie o 1 (DEC, DEX, DEY). Również wtedy, gdy wykonamy odejmowanie (tak! w języku asemblera jest to możliwe) i wynik wyjdzie nam mniejszy od zera. Również wtedy, gdy wynik np. dodawania zawarł się w przedziale od $80 do $FF. Wynika to jednak ze sposobu zapisu liczb ujemnych przez procesor do czego jeszcze dojdziemy.

V -o(V)erflow Flag - znacznik przepełnienia. Teoretycznie pojawia się w przypadku wykonywania operacji dodawania lub odejmowania, gdzie wynik końcowy przekroczy "granice przyzwoitości". W praktyce jednak korzysta się z tego bitu chyba tylko po wykonaniu rozkazu "BIT", który umieszcza weń zawartość szóstego bitu swojego argumentu. Oprócz tego istnieje jeszcze specjalna komenda "CLV" - (CL)ear o(V)erflow Flag, kasująca ten znacznik.

- - to właśnie ten nie wykorzystany bit, o którym wspominałem wyżej.

B - (B)reak Flag - znacznik programowego przerwania. Pojawia się gdy procesor wykonując program napotka komendę "BRK". BRK to skrót od (BR)ea(K), czyli "przerwij". W tym przypadku następuje zatrzymanie wykonywania programu i skok do odpowiednich procedur obsługi takiego przerwania. Standardowo wykonywane są te same procedury systemu operacyjnego, co w przypadku naciśnięcia kombinacji RUN/STOP - RESTORE.

D - (D)ecimal Mode Flag - znacznik dziesiętnego trybu pracy procesora. Znacznik ten przyjmuje wartość jeden, gdy procesor wykona komendę "SED". SED to (SE)t (D)ecimal Mode Flag, czyli ustaw znacznik D. W dziesiętnym trybie pracy procesor wykonuje wszystkie działania arytmetyczne w tzw. kodzie BCD. BCD oznacza (B)inary (C)oded (D)ecimal. A po polsku: system dziesiętny kodowany dwójkowo. Dla nas tryb ten będzie bardzo mało istotny gdyż wszyscy doskonale posługujemy się szesnastkowym sposobem liczenia czyli heksem. Mam nadzieję oczywiście! (Patrz również: KEBAB 1/92). Oprócz komendy "SED" istnieje również komenda o przeciwnym działaniu tj. kasująca znacznik D komenda CLD - (CL)ear (D)ecimal Mode Flag.-

I - (I)nterrupt Disable Flag - znacznik zabronienia obsługi przerwań. Jego pojawienie się jest warunkowane wykonaniem rozkazu "SEI" czyli (SE)t (I)... Flag.

W przypadku gdy znacznik ten przyjmuje wartość jeden, procesor ignoruje wszystkie sygnały dopraszające się o przerwanie. Zwracam jeszcze raz uwagę na nazwę tego znacznika. Nie jest to,

jak się mylnie spotyka w polskiej literaturze, "znacznik przerwań" co sugerowałoby, że wartość jeden oznacza przyzwolenie na przerwania tylko znacznik ZABRONIENIA obsługi przerwań (ang. disable = uniemożliwić, wyłączyć). Drugim przypadkiem kiedy wartość tego znacznika wynosi jeden, jest sytuacja kiedy procesor rozpoczyna wykonywanie przerwania. W tej sytuacji sam 6510 wstawia doń wartość 1 aby nie zaistniała taka sytuacja, że w trakcie wykonywania jednego przerwania nastąpi kolejne przerwanie i następne itd. Aby skasować znacznik (tj. przywolić na obsługę przerwań) mamy specjalną komendę "CLI" czyli (CL)ear (I)... Flag. Inna sytuacja kasująca znacznik I to powrót procesora do programu głównego po wykonaniu przerwania. O przerwaniach w C-64 będziemy dopiero mówili w następnych odcinkach, a wszystkich zainteresowanych samą istotą zjawiska (?!) odsyłam do artykułu o przerwaniach na Amigę (KEBAB 7-8/92).

Z - (Z)ero Flag - znacznik wartości zero. Znacznik ten to jeden z dwóch podstawowych znaczników wykorzystywanych przy szacowaniu wyników wszelkich działań arytmetycznych wykonywanych przez procesor. Jeżeli wykonamy jakąś operację, której wynikiem będzie zero to znacznik ten przyjmie wartość jeden. Kilka przykładów, w których znakiem "<-" zaznaczyliśmy rozkaz, po wykonaniu którego nastąpi ustawienie znacznika Z na jeden.

```
LDA #$00
```
<- w tym momencie pojawi się znacznik Z ...

```
LDX #$00        <- ...
LDX #$FF INX    <- ...
LDY #$01 DEY    <- ...
LDA #$03
STA $9000
DEC $9000
DEC $9000
DEC $9000       <- ...
```

itd.

Znane nam wszystkim (już!) komendy BEQ czy BNE korzystają właśnie z tego znacznika aby podjąć odpowiednią decyzję warunkową. Również porównania wykonywane przy użyciu komend CMP, CPX czy CPY sprowadzają się w efekcie końcowym do odpowiedniego ustawienia (1) bądź skasowania (0) znacznika Z. "Jak to?!" Ktoś mógłby zapytać. Jak wynik porównania może wynosić lub nie wynosić zero? Spójrzmy na to z nieco innej strony. Porównując dwie liczby zadajemy sobie pytanie: Czy te dwie liczby są równe? Na tak sformułowane pytanie możemy uzyskać tylko jedną z dwóch odpowiedzi: TAK lub NIE. Jeżeli teraz przyjmiemy, że odpowiedzi TAK odpowiada logiczna jedynka a odpowiedzi NIE logiczne zero, to mamy już układ zero-jedynkowy. Teraz wystarczy jeszcze tylko przyjąć, że znacznikiem odpowiedzialnym za odpowiedź na nasze pytanie jest znacznik Z rejestru SR procesora i już wiemy dlaczego komendy porównań (CMP etc.) ustawiają lub kasują ten właśnie znacznik. To jest wytłumaczenie nazwijmy je logiczne. Istnieje jednak jeszcze jedno wytłumaczenie bardziej matematyczne. Zadajmy sobie inne pytanie. Jak zrealizować matematycznie porównanie dwóch liczb? Otóż najprostszym wydaje się być odjęcie jednej od drugiej. Co nam to da? Otóż zawsze gdy obie liczby będą sobie równe, w wyniku odejmowania uzyskamy... **ZERO**! Jeżeli liczby będą różne to możemy uzyskać rozmaite wartości (dodatnie lub ujemne)ale na pewno **NIE** zero. I to jest cała filozofia! Procesor wykonuje dokładnie to samo gdy napotka rozkaz np. *CMP* tyle tylko, że rezultatu tj. właściwej różnicy nigdzie nie umieszcza ustawia jedynie lub kasuje (zależnie od rezultatu odejmowania) znacznik Z.

C - (C)arry Flag - znacznik przeniesienia. Zanim dokładnie omówimy ten znacznik musimy wytłumaczyć sobie jeszcze kilka spraw.

Wiemy już, że korzystając z języka asemblera, możemy wykonywać działania arytmetyczne. Dodawać, odejmować oraz (tego możemy jeszcze nie wiedzieć) mnożyć przez dwa i dzielić przez dwa. Jak jednak wykonujemy to w praktyce? Spróbujmy jak zwykle przedstawić to na przykładzie. Wyobraźmy sobie, że chcemy wykonać podstawowe działanie 2+2=? I napiszmy program, który nam to zrealizuje.

W BASIC'u napiszemy po prostu:

```
? 2+2
```

W AL musimy napisać trochę więcej:

```
LDA #$02
CLC
ADC #$02
```



W tym momencie mamy już w akumulatorze wynik naszego dodawania. Możemy jeszcze i zazwyczaj tak robimy, umieścić ten wynik w jakimś bezpiecznym miejscu pamięci bo akumulator zapewne będzie jeszcze nam potrzebny:

`STA $????` - gdzieś gdzie chcemy przechować wynik.

Co myśmy tu napisali? LDA #$02 to znany nam doskonale rozkaz umieszczający w akumulatorze wartość *2* ($02). Teraz następuje tajemnicze *CLC*. Uważni czytelnicy może już skojarzyli sobie tą komendę z wspominanymi wcześniej komendami *CLD*, *CLI* czy *CLV*.

Tak, ta komenda to właśnie kasowanie (nadanie wartości zero) znacznika C. Po co tak robimy przekonamy się za moment. Następna komenda to ADC #$02. Jak zawsze *ADC* to skrót od jakiegoś angielskiego zwrotu. Tym razem twórcy AL mieli na myśli zwrot: (AD)d with (C)arry czyli *Dodaj razem z przeniesieniem*. W praktyce *przeniesienie* oznacza wartość znacznika C. Zatem komenda ADC #$02 dodaje nam do zawartości akumulatora wartość $02 oraz wartość znacznika przeniesienia (C).

Teraz już wiemy po co była nam komenda *CLC*. Po prostu musimy być pewni, że w momencie dodawania znacznik C ma wartość zero.

W przeciwnym wypadku mogłoby nam wyjść, że 2+2=5... No dobrze, ale po co wogóle ten znacznik skoro i tak go kasujemy przed każdym dodawaniem? Wyobraźmy sobie nieco inną sytuację. Mamy np. wartość $FD i chcemy dodać do niej wartość np. $08. Ponieważ tak proste liczby potrafimy liczyć w pamięci, więc wiemy doskonale, że $FD+$08=$105. Tylko, że nasz program na dodawanie zamiast wartości $105 podaje nam w akumulatorze $05... No bo w sumie jak inaczej skoro akumulator jest rejestrem ośmiobitowym więc maksymalna wartość jaką może przechowywać wynosi tylko $FF.

Coś jednak należało zrobić aby problem ten rozwiązać. Wypróbujmy taki program:

```
A 5000 LDA #$FD
A 5002 CLC
A 5003 ADC #$08
A 5005 STA $6000
A 5008 BCC $5015
A 500A LDA #$01
A 500C STA $6001
A 500F LDA #$02
A 5011 STA $D020
A 5014 RTS
A 5015 LDA #$00 z
A 5017 STA $6001
A 501A LDA #$05
A 501C STA $D020
A 501F RTS
```

Po uruchomieniu np. "G5000" wykonajmy

```
M 6000
:6000 05 01 .. .. .. .. .. ..
```

Przy okazji ramka zmieniła nam się na czerwoną. Widzimy, że pod adresem $6000/$6001 mamy wartości $05 i $01. To jest właśnie nasz poszukiwany wynik.

Jeżeli teraz "złożymy" te dwie wartości (poczynając od drugiej) to otrzymamy poszukiwane $0105. Zróbmy teraz mały eksperyment.

Zmieńmy jedną linię w naszym programie i napiszmy:

```
A 5003 ADC #$02
G5000
M6000
:6000 FF 00 .. .. .. .. .. ..
```

No i zgadza się! bo $FD+$02=$FF lub $00FF jak kto woli. Przy okazji ramka ekranu przyjęła barwę zieloną. Skąd się to bierze? Otóż w przypadku, gdy wynik dodawania przekroczy maksymalną wartość, jaką można zapisać w akumulatorze, wówczas wykorzystuje znacznik przeniesienia (C) jako dodatkowy ósmy (bo liczymy od zerowego) bit. Daje nam to możliwość swobodnego dodawania dwóch liczb w zakresie od $00 do $FF. Zainteresowanym proponuję sprawdzić wpisując różne wartości w linie

```
A 5000 LDA #$??
i
A 5003 ADC #$??
```

A skąd my wiemy jaki jest stan wskaźnika C? Pewnie Czytelnicy zauważyli już jedną dziwną komendę w naszym programie.

BCC - to właśnie jest to magiczne słowo. Oznacza ono *(B)ranch on (C)arry (C)lear* czyli *skocz jeżeli znacznik C jest skasowany.* Po prostu procesor wykonując tą komendę sam sprawdzi sobie jaki jest stan znacznika C i podejmie odpowiednią decyzję. Istnieje jeszcze inna komenda odnosząca się do tego samego znacznika.

Komenda ta to *BCS*. Jest to akurat przeciwna komenda do *BCC*. Podobnie jak mieliśmy w przypadku *BEQ* i *BNE* tak i tu *BCS* oznacza *(B)ranch on (C)arry (S)et* czyli *skocz gdy znacznik C jest ustawiony.*

Proponuję wszystkim Czytelnikom napisanie naszego programu na dodawanie ale z wykorzystaniem komendy *BCS*.

W następnym odcinku zajmiemy się jeszcze przez chwilę działaniami arytmetycznymi (konkretnie odejmowaniem) gdyż będzie nam to bardzo przydatne w przyszłości oraz napiszemy (wreszcie!) naszego pierwszego scroll'a. Zapraszam!

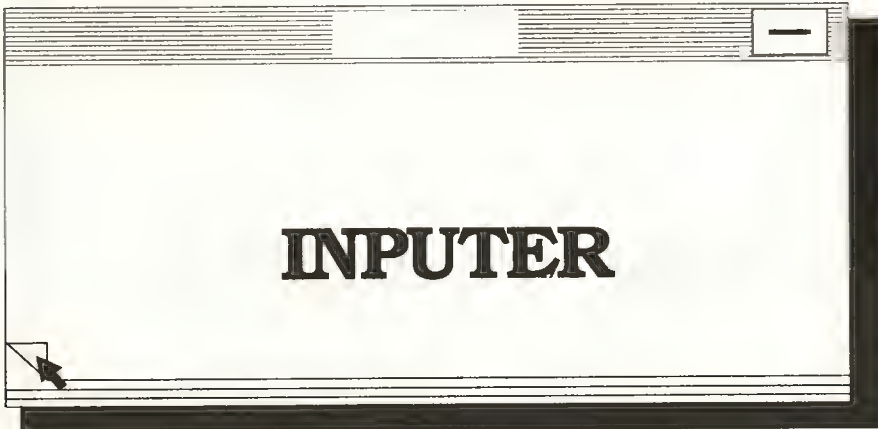
SD!

Słowniczek:

Flag - to po angielsku *"coś co określa nam, że coś się wydarzyło lub nie" (C-64 Programmer's Reference Guide p.214).* Hmm... "Coś, że coś i coś..." nie podoba mi się takie tłumaczenie. Spróbujmy to powiedzieć jakoś bardziej "po ludzku". Spójrzmy na naszego Commodorka. Zależnie od tego, czy mamy starszą czy nowszą wersję obudowy, mamy nań z prawej lub lewej strony umieszczoną diodę świecącą (LED). Oczywiście jest ona świecąca tylko wtedy gdy mamy włączone zasilanie. I to właśnie może być przykład naszego "flaga". Istnieją tylko dwie możliwości: świeci lub nie. Inny przykład. W każdym prawie samochodzie, oprócz rozmaitych *zegarów* znajduje się kilka (lub kilkanaście) rozmaitych *kontrolek* czyli lampek informujących kierowcę o stanie rozmaitych urządzeń samochodu. Jeżeli zaczyna brakować paliwa włącza się odpowiednia lampka i informuje o tym kierowcę. Jeżeli ciśnienie oleju w silniku spadnie poniżej dopuszczalnego poziomu czy włączymy światła, kierunkowskazy etc... również odpowiednia lampka zaświeci się na desce rozdzielczej. Każda z tych lampek to *flag* określający stan jakiegoś obwodu lub podzespołu w samochodzie. W przypadku procesora deską rozdzielczą jest rejestr stanów *(SR)* a kontrolki to poszczególne bity odpowiadające za rozmaite *podzespoły* procesora. Jak jednak nazwać to coś po polsku? Najbardziej odpowiednim słowem wydaje się słowo: znacznik.

LSB - to skrót od angielskiego *Least Significant Bit.* Po polsku oznacza to najmniej znaczący bit. Oczywiście *najmniej znaczący* nie odnosi się do jego ważności dla procesora czy komputera tylko oznacza wpływ tego bitu na wartość całego bajtu. Jak zapewne wiemy bit zerowy nazywa się tak dlatego, że jego wpływ na wartość bajtu wynosi zero lub dwa do potęgi zerowej. Bit siódmy "wnosi" do wartości bajtu zero lub dwa do potęgi siódmej. Wpływ więc bitu numer zero jest wielokrotnie mniejszy niż pozostałych - stąd nazwa *LSB*

MSB - to skrót od *Most Significant Bit.* Zapewne się już domyślamy, że tym razem chodzi o coś zupełnie przeciwnego niż *LSB*. Tak! *MSB* to najwięcej znaczący bit. W przypadku jednego bajtu, najwięcej znaczącym bitem jest bit numer siedem. W przypadku rejestrów o większej niż osiem ilości bitów *MSB* to ostatni z bitów danego rejestru. Dla przykładu rejestr *PC* w *6510* jest rejestrem 16-o bitowym a jego *MSB* to bit numer 15.



INPUTER

Po wielu niepokojach i przeciwnościach losu udało nam się wreszcie wprowadzić w KEBABIE standard publikowania wydruków programów w kodzie maszynowym w taki sam sposób, jak zwykle drukują zawartość pamięci programy monitorujące.

Dodatkowe sumy kontrolne pozwalały na korekcje błedów przy pomocy programu KOREKTOR. Wszystko to było z założenia proste i przejrzyste choć nie pozbawione jednej, oczywistej wady - wydruki najprostszych nawet programów były dość długie jak na objętość naszego miesięcznika. Skazywało to nas (Redakcję) do zamieszczania tylko i wyłącznie bardzo krótkich programów, bo na większe po prostu nie było miejsca.

Po parudniowych konsultacjach zrodził się w końcu pomysł całkowicie nowego systemu publikowania programów na łamach KEBABA. Opracowany system pozwala opublikować wydruki programów, które podane w tej formie są o ponad 1/3 krótsze (!) od ich odpowiedników wydrukowanych w systemie KOREKTOR. Dodatkowo inny układ znaków w wierszach pozwala na bardziej racjonalne wykorzystanie powierzchni strony, co niesie następne oszczędności.

Jak to działa?

Otóż w przypadku wydruków w systemie KOREKTOR wszystkie dane były drukowane w postaci liczb szesnastkowych, a więc np. A0 BD 02 1E itp. Jak nietrudno zauważyć, każdemu rzeczywistemu bajtowi odpowiadały 2 znaki.

W systemie INPUTER zastosowano kod złożony z cyfr, liter i znaków specjalnych, gdzie na jeden znak w wydruku przypada 0.75 rzeczywistego bajtu. Ponieważ może się zdarzyć, że znak 0 (zero) i litera O wyglądać mogą tak samo, zdecydowano się zrezygnować całkowicie z używania tejże litery.

Tak więc znaki, co do których Czytelnicy mogliby mieć wątpliwości, czy jest to zero czy litera O, są na pewno tylko i wyłącznie znakami ZERA.

Paniatno? Dodatkowo program do wprowadzania listingów w tym formacie zawiera testowanie klawiatury i znaki nie objęte w definicji kodu INPUTER po prostu nie ukażą sie na ekranie.

Program pracuje w języku maszynowym a Czytelnicy powinni wpisać jego napisany w BASIC ładowacz (brzydko brzmi, ale wszyscy wiedzą, o co chodzi). Po jego uruchomieniu czekamy sobie około 1 minuty (jest tych danych trochę...) i po wybraniu wersji dla taśmy lub dyskietki możemy rozpocząć wpisywanie listingów.

Po przystąpieniu do pracy INPUTER jest od razu ustawiony na wpisywanie programów począwszy od adresu $0801, a tam właśnie zaczynać się będzie większość naszych programów. Jeżeli trzeba będzie ów adres zmodyfikować, wystarczy wcisnąć F-1 i podać nowy adres.

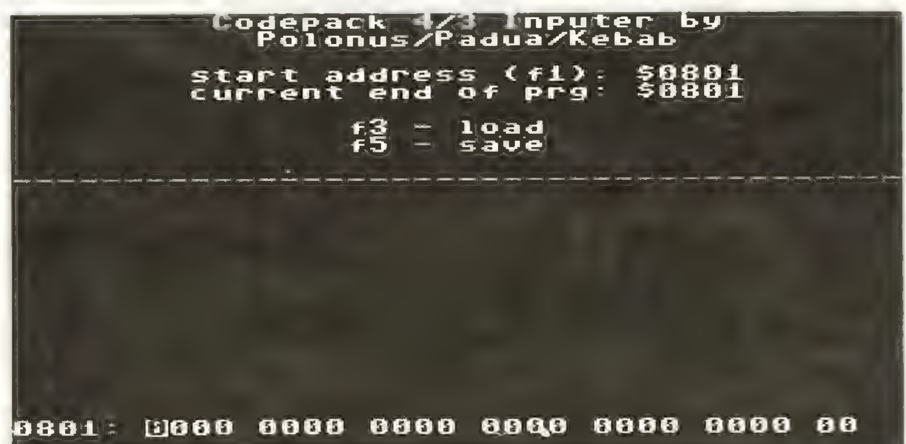
Jak zbudowany jest kod, którym posługuje się INPUTER? Dla przykładu obejrzyjmy sobie pierwszą linię opublikowanej w tym samym numerze gry, której listing wydrukowano właśnie w standardzie INPUTER:

```
0801 2Myg 1FU! c3gV 0a00 ue&1
KtA& 63
```

"0801" to oczywiście adres, począwszy od którego wpisywane będą dane z tej linii. Ma on charakter wyłącznie informacyjny, gdyż jego drukowaniem na ekranie oraz zwiększaniem zajmuje się automatycznie INPUTER. Następujące po nim 6 czteroznakowych grup zawiera opis 18 bajtów wpisywanego programu. Obecna na końcu liczba 63 to właściwa dla tej linii suma kontrolna. Po właściwym przepisaniu każdej z takich linii ukazywać się

Do Zapamiętania!

- Rejestr *SR*
- znaczniki *N, V, B, D, I, Z, C*
- komendy czyszczące i ustawiające znaczniki (*CL?* i *SE?*)
- komenda *ADC*
- komendy skoków warunkowych (*BCC, BCS*)



będzie na ekranie znaczek "+" (zaraz za sumą kontrolną). Oznaczać on będzie, że możemy teraz wcisnąć RETURN i zająć się wpisywaniem następnej linii. Należy pamiętać, że to dopiero wciśnięcie RETURN zapisuje poprawną linię w pamięci. Jest to ważne przy operacjach LOAD i SAVE.

INPUTER został napisany tak, aby można było przerwać wpisywanie w dowolnym momencie i nagrać dane na nasz nośnik celem poźniejszej kontynuacji wpisywania. Po wykonaniu opcji LOAD (klawisz F-3) wpisywanie będzie kontynuowane od ostatniego wpisanego wiersza programu.

Paweł "Polonus" Sołtysiński

Mini - Penetrator

Dzisiaj niespodzianka dla tych, którzy lubią sobie trochę pograć (zakładając oczywiście, że pogodzą się z koniecznością wpisania programu). Sama idea gry jest dość prosta a nadany jej tytuł ("Mini-Penetrator") nawiązuje do gry, która była wydana parę lat temu - ot, "leci się i strzela, do czego popadnie". Gra miała jednak coś sympatycznego i po prostu wciągała.

Od pewnego czasu zastanawiałem się nad zrobieniem jakiejś gry, którą można by było opublikować w miejsce zwykle bardzo tematycznie poważnych programów użytkowych. Dla większości z nas komputer jest narzędziem zabawy i tak powinno być. Problemem było tylko zapisanie w pamięci kształtów trasy, na której toczy się gra.

Z oczywistych przyczyn program nie może być długi, więc postanowiłem zrezygnować z opisu tej trasy biorąc za jej opis ...dane umieszczone w pamięci ROM komputera (dla dociekliwych: sklepienie groty to dane z Kernal ($E000-$FFFF) a dno to interpreter Basic ($A000-$C000).

Program należy wpisać korzystając z zamieszczonego w tym numerze Kebaba programu korygującego **INPUTER.** Po wpisaniu program należy nagrać na dyskietkę lub taśmę. Gotową grę uruchomić możemy przy pomocy dyrektywy RUN.

Dla zapalonych graczy mała informacja: w grze umieściłem tzw. cheat, czyli sposób na włączenie nieśmiertelności, a dokładniej - na podniesienie ilości "żyć".

Gra powstała z wykorzystaniem jednego generatora znaków, który zawiera przedefiniowany zestaw liter oraz dane do grafiki gry "właściwej".

Miłej zabawy życzy

Paweł "Polonus" Sołtysiński.







Basic cd.

W dzisiejszym odcinku naszych spotkań z językiem BASIC dla Commodore 64 poznamy kilka użytecznych technik, których znajomość czyni dopiero z nas zręcznego programistę. Większość z nich polega na modyfikacji tzw. zmiennych systemowych co w efekcie powoduje zmiany w pracy komputera. Modyfikacji dokonujemy przez użycie rozkazu POKE, który w języku BASIC pozwala na wpisanie do podanej lokacji pamięci danej wartości jednobajtowej. Format rozkazu wygląda następująco:

POKE adres,wartość

Adresem może być liczba lub wyrażenie, które w wyniku da nam liczbę z zakresu od 0 do 65535 (czyli adresy od $0000 do $FFFF w zapisie szesnastkowym). Wartość to liczba (lub, analogicznie, wyrażenie) o możliwych wartościach od 0 do 255. Obie liczby (adres i wartość) powinny być liczbami całkowitymi. Odwrotnością jest funkcja PEEK, która przyporządkowuje zmiennej zawartość podanej komórki. Przykład:

PRINT PEEK(53280) - drukuje na ekranie wartość odpowiadającą kolorowi ramki.

A oto kilka przykładów wraz z opisami:

POKE 808,215 - wyłącza możliwość zatrzymania programu w BASIC poprzez wciśnięcie RUN/STOP;

POKE 774,134:POKE 775,227 - wyłącza działanie komendy LIST (po wykonaniu podanych POKE'ów LIST jest niemożliwy);

POKE 19,16 - usuwa znak zapytania, jaki pojawia się przy wywołaniu instrukcji INPUT. POKE 19,0 przywraca standardowe ustawienie;

POKE 53265,PEEK(53265)AND239 - wyłącza ekran (wszytko "staje się" ramką), co ma szczególne znaczenie przy dłuższym okresie oczekiwania np. na wyniki obliczeń. Jest to zasadne, gdyż z wyłączonym ekranem procesor pracuje o 16 procent szybciej niż z włączonym;

POKE 53265,PEEK(53265)OR16 - włącza ekran ponownie;

POKE 56334,0 - dodatkowe przyspieszenie pracy poprzez wyłączenie przerwań. Należy jednak pamiętać, że działalność klawiatury jest wówczas zawieszona.

POKE 56334,8 - ponowne włączenie przerwań;

POKE 650,128 - powoduje samopowtarzanie wszystkich klawiszy (po przyciśnięciu i przytrzymaniu);

PRINT PEEK(43)+PEEK(44)*256 - drukuje adres początku programu w j.BASIC (przeważnie 2049);

PRINT PEEK(45)*PEEK(46)*256 - drukuje adres końca programu w BASIC;

POKE 646,kolor - ustawienie aktualnego koloru kursora (wartości od 0 do 15);

Przy wykorzystaniu komendy PEEK() można odczytać np. położenie joystick'a. Aby wyjaśnić zasadę tego odczytywania posłużmy się przykładem:

```
10 A=PEEK(56320)AND15:K=0:S=0
20 IF A=14 THEN K=1:REM GORA
30 IF A=6  THEN K=2:REM GORA+PRAWO
40 IF A=13 THEN K=3:REM PRAWO
50 IF A=13 THEN K=4:REM DOL
60 IF A=9  THEN K=5:REM LEWO+DOL
70 IF A=11 THEN K=6:REM LEWO
80 IF A=10 THEN K=7:REM GORA+LEWO
90 IF (PEEK(56321)AND16)=0 THEN S=1:REM STRZAL
```

Przedstawiony wyżej krótki programik pozwala na przetestowanie wychylenia joystick'a (w porcie numer 2) oraz na zauważenie przyciśnięcia przycisku strzału. W przypadku braku ruchu (położenie pionowe) oraz strzału, zmienne K i S są ustawione na zero.

Aby uzyskać podobne dane o joystck'u w porcie 1, należy zamienić adresy w instrukcjach PEEK (z linii 10 i 90) z 56320 na 56321.

Posługując się instrukcjami POKE i PEEK można wykonać we własnym zakresie procedry zastępujące również inne brakujące funkcje. Jednym z przykładów może być podprogram szybkiego odczytu znaku z

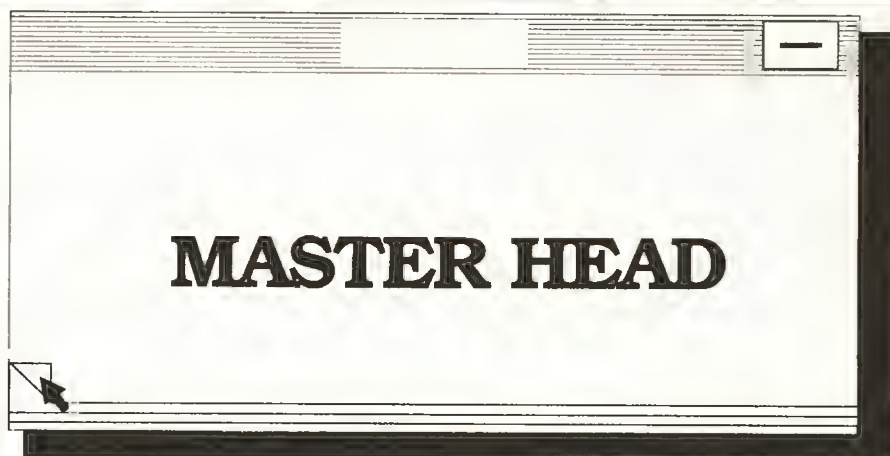
```
10AD=1024+X+Y*40:ZNAK=PEEK(AD):KOLOR=PEEK(AD+54272)AND15
```

ekranu. Funkcja ta występuje np. w ZX Spectrum jako SCREEN, gdzie odczytuje się kod znaku w określonym miejscu ekranu wraz z odpowiadającym mu numerem koloru. Można to wykorzystywać np. przy pisaniu własnych gier itp.

Wykonanie takiej linii programu (jej numer należy oczywiście zmienić stosownie do potrzeb) zwróci nam kod ekranowy znaku (o ustalonych poprzednio współrzędnych X i Y) w zmiennej ZNAK oraz numer koloru atrybutu w zmiennej KOLOR. Współrzędne insteresującego nas położenia na ekranie należy wpisać do użytych tu zmiennych X i Y.

I to by było tyle w tym, chyba już ostatnim artykule z tego cyklu. Od następnego, miesiąca - wydruki gotowych programów wraz z opisami. Najlepsza nauka programowania to oglądanie i rozumienie czyichś programów.

Paweł Sołtysiński



Tego jeszcze nie było! KEBAB dostał do przetestowania grę napisaną w Polsce! Dzięki uprzejmości producenta - spółki "Sonix" ze Szczecina (a jakże, wszyscy pod ręką...) do naszego redakcyjnego C-64 trafiła dyskietka i kaseta z grą o nazwie Master Head. Zarówno dyskietka jak i kaseta posiadają firmowe koszulki, na których uwidoczniono tytuł gry, producenta oraz bohatera gry.

Z całym szacunkiem dla przedsięwzięcia ale projektującego obrazek grafika to ja bym już sam nie wiem co! Krótko mówiąc, bohater od razu skojarzył mi się ze zzieleniałą od niewłaściwego przechowywania pyzą w czerwonych buciorkach.

Ale niech tam... W końcu to ma tylko przyciągać(?) klienta. Po wgraniu wersji taśmowej (wgrała się bezproblemowo już po pierwszym ustawieniu głowicy) oczom moim ukazała się strona tytułowa.

W sumie sympatyczna grafika (pyza z koszulki też jest), możliwość wyboru instrukcji lub rozpoczęcia gry. Po wybraniu instrukcji okazuje się, że jako nowy pomocnik w magazynie przyjdzie nam rozmieszczać paczki we wskazanych miejscach. Zadanie wydaje się z pozoru proste, wobec tego wracam do ekranu tytułowego i wybieram start gry.



Na dole ekranu ukazał się panel informacyjny (ilość prób, czas itp.) a u gory - pole gry. W każdej ze stref jest to niewielki labiryncik (widocznie pomieszczenia magazynowe są z odzysku po dawnych schronach wojskowych) wraz z naszym bohaterem pośrodku plus kilka paczek i oznaczonych dla nich miejsc. Paczki można przesuwać. Przesunięta paczka może zablokować inną paczkę. Zablokowana paczka nie dałę się przesunąć w ogóle, nie mówiąc już o umieszczeniu jej na miejscu. Czyli trzeba najpierw pomyśleć a potem się pchać (jak w życiu!)

Parę nieudanych prób i wymowny obrazek informuje nas, że takiego pracownika to w tym magazynie nie potrzebują. Trudno, zapuszczamy wąsy, farbujemy włosy i nierozpoznani próbujemy znowu. W sumie bardzo sympatyczna gra i rzeczywiście wymagająca trochę myślenia.

Na koniec trochę danych o autorach:

Twórcami pierwowzoru (chyba

BASIC - Protector

No cóż, Czytelnik nasz Pan, tak więc program ten otwiera rubrykę programików na zamówienie, tzn. dział, w którym publikowane będą krótkie procedury, których wykonanie było sugerowane przez samych Czytelników. Na pierwszy ogień - program do zabezpieczania programów w języku BASIC.

Czego potrzebowało kilku naszych Czytelników? Problem wyglądał następująco: potrzebowano procedury, która uniemożliwiła by wylistowanie treści programu w BASIC. Prezentowany "zabezpieczacz" robi to skutecznie (linia z dyrektywą SYS) oraz dodatkowo blokuje możliwość przerwania pracy programu w BASIC przy pomocy klawisza RUN/STOP.

Jakie są zasady użycia programu Protector? Proste: program w BASIC powinien być dłuższy niż 256 bajtów oraz pozbawiony błędów programowych, tak, aby jego wykonanie nie zostało przerwane przez "Syntax Error in ....".

Po uruchomieniu wpisanego uprzednio programu należy wgrać program przeznaczony do zabezpieczenia (ale nie uruchamiać go!) a następnie wywołać zabezpieczanie przez rozkaz SYS 49152.

Po tej operacji zabezpieczoną wersję należy nagrać na dyskietkę lub taśmę. I to wszystko.

Jeżeli któryś z naszych Czytelników ma jakiś problem, z którym nie może sobie samodzielnie poradzić, wystarczy telefonicznie lub listownie powiadomić o tym dział C-64 w redakcji Kebaba.

Paweł Sołtysiński

dla małego Atari) byli panowie Tomasz Liebich, Radosław Buchta, Michał Widera i Grzegorz Pancherz. Autorem wersji dla C-64 był Zenon Mikołajczyk (dla Atarowian - Magnusoft).

Wiele pomyślności w imieniu KEBABA życzy

Paweł Sołtysiński



Czy wiedzieliście, że...

przy komendzie ***LoadWB*** wykonywanej np. z poziomu startup-sequence naszego dysku systemowego możemy dodać parametr *-debug?* Co nam to da? Zarówno w przypadku workbench'a 1.3 jak i 2.0 doda nam jeszcze jedno menu systemowe. W menu tym znajdziemy w zależności od wersji systemu opcję *debug* lub *ROMWack oraz* wspólną opcję *flushlibs.*

O ile pierwsza opcja powoduje na pierwszy rzut oka nic więcej niż "zawieszenie" *się* systemu o tyle drugą mogą się zainteresować wszyscy, którym nie podoba się to, że po uruchomieniu np. CED'a i powróceniu z powrotem do Workbench'a. mamy znacznie mniej pamięci niż przedtem. Czemu tak jest? Otóż załadowane przy uruchomieniu programu biblioteki (ale również fonty, urządzenia itp...) system pozostawia mimo zamknięcia (CLOSE) dalej w pamięci.

Umożliwia to szybszy start kolejnych programów (lub tego samego) wykorzystujących biblioteki itp. Gdyż nie muszą one być ponownie odczytywane z dysku jeżeli jednak użyjemy flushlibs zarezerwowana pamięć zostanie zwolniona.

Opcja *debug (ROMWack)* powoduje zamrożenie całego systemu. Po takim zatrzymaniu wszystkich programów można przy użyciu specjalnego programu komunikacyjnego odczytać poprzez serial port. stan np. wszystkich rejestrów procesora oraz wiele innych interesujących dla programisty wiadomości.

COPY - PARTY!!!

Party odbędzie się w dniach 6-8 listopada w szkole nr. 220 przy ulicy Jana Pawła II, zaraz przy giełdzie na Grzybowskiej (po drugiej stronie Jana Pawła II).

Co będzie na Party?

1. Będą identyfikatory.
2. Będzie elita, która otrzyma uprzywilejowane miejsca.
3. Będzie duże pomieszczenie na competition i jeden lub dwa bigscreen'y.
4. Organizatorzy gwarantują wysokiej jakości nagłośnienie oraz własną stację telewizyjną, nadającą nonstop informacje o party i competitions.
5. Będzie trochę szybkich modemów (HST) i świeże importy.
6. Będzie koncert zespołu "Root and Instruments" *****Italic*****
7. Będzie Fast-Food, chipsy i napoje chłodzące.
8. Będzie trochę lamerstwa z którego można się będzie pośmiać.
9. Będzie pizza na telefon.
10. Będzie cool...

Competitions.

Organizatorzy przewidują różnego rodzaju konkursy, a przede wszystkim trzy główne:

- graphics (I nagroda - 1.000.000 zł)
- music (I nagroda - 1.000.000 zł)
- demo (I nagroda - 2.000.000 zł)

W zależności od chętnych i nastroju mogą wystąpić konkursy w innych kategoriach.

Zasady Demo competition.

1. Dyskietka musi być podpisana.
2. Demo musi chodzić na podstawowym typie Amigi. Jeżeli demo wymaga 1MB Chip, trzeba to zaznaczyć.
3. Demo musi być oryginalne - demomakery odpadają.
4. Demo musi zostać oddane do konkursu na dwie godziny przed jego rozpoczęciem (konkursu oczywiście).

Zasady Music competition.

1. Do konkursu można oddać maksymalnie 1 moduł. Spośród oddanych zostaną wyselekcjonowane przez kompetentne jury, najlepsze produkty i te dopiero zostaną dopuszczone do konkursu.
2. Dyskietka z modułem musi zawierać dane o autorze, grupie, czasie trwania, timing'u, programie muzycznym.
3. Moduł musi znajdować się w katalogu MODULES.
4. W tytule i nazwach sampli nie mogą znajdować się informacje o autorze.
5. Ostateczny termin oddania modułu do konkursu upływa w terminie 4 godzin przed rozpoczęciem.

Zasady Graphics competition.

1. Każdy może zaprezentować maksymalnie dwa obrazki.
2. Scaningi będą wyeliminowane.
3. Dyskietka z pracami musi zawierać dane o autorze, grupie, tytułach, programie graficznym na którym obrazek został wykonany.
4. Obrazki muszą znajdować się w katalogu głównym.
5. Ostateczny termin oddania obrazków do konkursu upływa w terminie 2 godzin przed rozpoczęciem.

Plan akcji.

1. Party rozpoczyna się w piątek o 17.00. Osoby nie związane z organizacją, które pojawią się wcześniej pocałują klamkę.
2. Pierwszy konkurs (music) odbędzie się w sobotę o 16.00.
3. Graphics competition odbędzie się w sobotę o 20.00.
4. Demo competition odbędzie się w sobotę o 23.00.
5. Wyniki zostaną podane w niedzielę o 12.00.
6. Koniec party o godzinie 17 w niedzielę.

Wstęp na party wynosi 40.000 zł od osoby. Przy wejściu każda osoba dostanie kartę do głosowania i identyfikator. Organizatorzy proszą o zabranie śpiworów. Każda grupa proszona jest o przysłanie potwierdzenia przyjazdu. Istnieje możliwość wynajęcia osobnej sali - koszt 50.000zł.

Od redakcji:

Powyższa notatka sporządzona została na podstawie zaproszenia rozsyłanego przez organizatorów.

Jak widać Party zapowiada się niezmiernie zachęcająco i na razie wydaje się być najbardziej profesjonalnie przygotowaną tego typu imprezą spośród tych, które do tej pory się odbyły. Oczywiście Kebab również będzie tam obecny, więc będziecie mogli przeczytać na naszych łamach relację z tej imprezy.

ANKIETA

1. Jaki typ komputera Commodore posiadasz i z jakim osprzętem.

....

2. Czy zamierzasz w przyszłości dokupić jakieś wyposażenie dodatkowe?
Jeśli tak to jakie?

....

3. W jaki sposób używasz przeważnie swojego komputera?
Nadaj ocenę wskali od 1 do 5 punktów każdej z pozycji.

a) Gram. ...
b) Programuję. ...
W jakim języku? ....
c) Używam do pracy. ...
d) Używam do nauki. ...
e) Rysuję lub komponuję. ...
f) Inne...

Czy wiedzieliście, że...

... istnieje stary jak C-64 trik uniemożliwiający wylistowanie programu w BASIC'u za pomocą prostej komendy "LIST"?

Co należy zrobić? Wystarczy aby pierwsza linia naszego programu zawierała komendę "REM" z następującym po niej kodem odpowiadającym znakowi (literze) SHIFT-L. Próba wylistowania takiej linii kończy się, w przypadku normalnego systemu operacyjnego, lakonicznym komunikatem:

?SYNTAX ERROR

SD!

(Jakie?) ....

4. Jakiego typu gry najbardziej Ci odpowiadają?

a) Shoot'em UP (Strzelanki; bij-zabij)
b) Adventure (Przygodowe)
c) Strategic (Strategiczne)
d) Symulation (Symulacyjne)

5. Podaj swoje trzy ulubione gry.

....

6. Podaj trzy, najbardziej według Ciebie udane programy użytkowe.

....

7. Od ktorego numeru czytasz Kebab? ......

8. Czy czytasz inne gazety o Amidze/C64 (64+4,C&A,Amigowiec)? Jeśli tak to jakie? Którą uważasz za najlepszą? Dlaczego?

....
....

9. Co Ci się podoba w Kebabie?

....
....

10. Co Ci się nie podoba w Kebabie?

....

....

11. Co nowego/więcej chciałbyś widzieć w Kebabie?

....
....
....

12. Jak zdobywasz Kebab?

a) Kupuję w kiosku
b) Pożyczam od kolegi
c) Prenumeruję
d) Kupuję na giełdzie
e) Inaczej (jak?) ....

21. Czy masz kłopoty ze zdobyciem Kebaba?

a) Tak, bardzo duże
b) Tak, często
c) Czasami
d) Nie mam kłopotów

23. Podaj swój wiek.

a) Mniej niż 14 lat
b) 14 do 25 lat
c) Ponad 25 lat

Imię i Nazwisko:
....

Dokładny adres zamieszkania:
....

Wszyscy którzy nadeślą ankiety wezmą udział w losowaniu prenumeraty Kebaba.

Każdy kto nadeśle ankietę zostanie wpisany do specjalnej bazy danych, na podstawie której będziemy (w miarę możliwości) rozsyłać różne atrakcyjne propozycje:

* Zaproszenia na imprezy;

* Adresy klubów komputerowych;

* Atrakcyjne oferty kupna programów i sprzętu komputerowego;

* Być może wiele, wiele innych...

Amiga 4000

Dzięki uprzejmości Biura Przedstawicielskiego firmy "Commodore" uzyskaliśmy informacje "z pierwszej ręki" dotyczące najnowszego produktu z serii Amiga. Wydaje się, że po całej serii mniej lub bardziej nieudanych (naszym zdaniem) kroków, firma Commodore stworzyła nareszcie coś na co od dawna czekaliśmy... Amiga 4000!

W podstawowej konfiguracji będzie to komputer wyposażony w procesor 68040 taktowany częstotliwością 25MHz (czemu nie 33?) wyposażony w nowy zestaw układów graficznych nazwany w skrócie "AA" co ma oznaczać: zaawansowana Amiga (Advanced Amiga). Cztery megabajty pamięci RAM z czego dwa to pamięć typu "Chip" należy również do podstawowego wyposażenia.

Na płycie głównej znajdują się trzy wolne slot'y typu Zorro III/AT-ISA oraz jeden typu Zorro III/Video. A4000 jest wyposażana w 40 (??!) lub 120-o megabajtowe napędy twardych dysków typu AT-Bus (czemu nie SCSI?). Napędy dysków elastycznych pozwalają na pracę z dyskietkami zapisanymi w formacie Amiga-DOS (880KB lub 1,76MB), oraz MS-DOS (720KB lub 1,44MB). AA - pozwala (nareszczie) na rozszerzenie podstawowej palety kolorów do 16,7 mln.

Niestety znowu nie zdecydowano się na konkretne uderzenie w bezpośredniego (już niestety) konkurenta jakim stał się szybki (?) PC z najnowszymi kartami graficznymi. Z palety można maksymalnie wybrać i pokazać na ekranie tylko 256 kolorów. Jedynie tryb HAM8 (ośmiobitowy HAM) pozwala na uzyskanie większej ich liczby. Również rozdzielczości typu 640x400 do 1280x512 nie są już dziś niczym rewelacyjnym.

Tak więc producenci rozszerzeń graficznych typu Impact Vision, Video Toaster czy, opisywany wstępnie w niniejszym numerze, Opal Vision, mają jak na razie spokojny sen. W dziedzinie fonii nowa Amiga nie prezentuje niczego szokującego. Cztery kanały ośmiobitowych sampli to znana już z Amigi 1000 (dla przypomnienia: rok 1985) Paula. Również i tu nie zdecydowano się na poważniejsze uderzenie w konkurencję (patrz np. Atari Falcon 030).

Wynika z tego, że i tu mają pole do popisu postronni producenci kart dźwiękowych np. typu "Maestro". Do nowego komputera stworzona została również nowa wersja systemu operacyjnego, który nazywa się aktualnie "Amiga-OS 3.0" Zgodnie z wypowiedziami niezależnych obserwatorów nazwa "3.0" wydaje się być jakby trochę na wyrost.

O ile różnice pomiędzy wersją 1.3 (1.4) a 2.0 były na tyle istotne, że w pełni zasługiwały na zmianę pierwszej cyferki numeru wersji, o tyle różnice pomiędzy wersjami 2.0 a 3.0 są zdecydowanie mniejsze (cyt. z dostarczonych materiałów informacyjnych).

Zdecydowanie na uwagę zasługuje fakt zainstalowania procesora głównego nie na płycie głównej (wśród panującej tam technologii SMT - patrz test A600 KEBAB 4/92 str.20) lecz na specjalnej karcie "daughter board" umieszczanej w przeznaczonym dlań gnieździe (CPU-Slot).

Rozwiązanie to umożliwia w przyszłości łatwą wymianę tejże karty na inną zawierającą np. szybszy procesor (choćby 33MHz) lub np. DSP (Atari Falcon 030) i otwiera drogę do "poprawienia" A4000 również innym producentom. Należy się spodziewać, że już wkrótce po wejściu "czterytysiączki" na rynek firma PP&S (Progressive Perifferals & Software) czy GVP (Great Valley Products) zaproponuje nam zapełnienie tego gniazda czymś innym niż standardowy 68040 - 25MHz. Oczywiście nie oznacza to, że A4000 jest komputerem powolnym.

Już A3000 lub A2000 z kartą np. "G-Force 030 50MHz" mogły stawać spokojnie do zawodów z każdym "pecetem". A4000 dzięki procesorowi nowszej generacji jest w większości testów blisko dwukrotnie szybsza niż A3000-25. Zastanawia natomiast upór Commodore w lansowaniu znanego z PC-kompatybilnych łącza twardego dysku typu AT-Bus.

O ile w przypadku A600 było to wybaczalne (w końcu to tylko najprostszy model) o tyle stosowanie tego nieco przestarzałego łącza w komputerze typu High-End czyli najbardziej zaawansowanym technicznie, wydaje się być co najmniej zastanawiające.



Dodatkowo przeciwko instalowaniu takiego rozwiązania przemawia fakt, że nawet producenci komputerów typu PC wycofują się powoli ze stosowania AT-Bus'a zastępując go znacznie bardziej przyszłościowym SCSI. Również niezależne firmy (GVP, PP&S, Supra i in.) produkują kontrolery SCSI dla wszystkich modeli Amigi. Mamy nadzieję (ciągle jeszcze), że sprawy te są tylko wynikiem chęci obniżenia kosztów i szybszego uruchomienia produkcji nowego modelu. Co by jednak nie mówić (pisać) A4000 jest nareszcie pierwszym poważnym krokiem Amigi w stronę świetlanej przyszłości.

SD!

Nowe gry na rynku na podstawie Amiga World (August 92)

Panzer Battles
(Strategic Studies Group)

Lato 1941. Operacja Barbarossa - nazistowskie Niemcy atakują Rosję rozpoczynając z nią blisko cztero-letnią wojnę. W serii bitew toczonych od przedmieść Moskwy do ulic Berlina, wojna na Rosyjskim froncie ostatecznie przypieczętowała los Trzeciej Rzeszy Hitlera. Panzer Battles firmy Strategic Studies Group przenosi nas w atmosferę tych wydarzeń odtwarzając sześć wielkich bitew pancernych.

Zamiast poruszania z osobna każdą jednostką po heksagonalnej mapie możesz (tak jak twój komputerowy lub żywy przeciwnik) pełnić rolę dowódcy oddziałów wydając jedynie rozkazy dowódcom poszczególnych pułków. Oddziały starać się będą wypełniać twoje rozkazy tak dokładnie jak to możliwe.

Jeśli poczujesz się zmęczony sześcioma wbudowanymi scenariuszami, możesz utworzyć własne! Możliwe jest zarówno wyrysowanie nowego terenu jak i jednostek przy pomocy narzędzi nazwanych Warplan i WarPaint

Jedną z męczących cech gry jest jej niskorozdzielczy ekran (320/200 32 kolory). Z tego powodu program ciągle musi zmieniać fragment pokazywanej mapy. Brak także tak przydatnej dużej mapy strategicznej co uniemożliwia zobaczenie w całości pola bitwy.

Panzer Battles nie jest zabezpieczony przed kopiowaniem, zawiera program także pozwalający w łatwy sposób zainstalować go na twardym dysku. Nie jest to jednak konieczne gdyż doskonale radzi sobie także na Amidze wyposażonej w dwie stacje dysków. W obu przypadkach wymagane jest 1MB pamięci RAM.

Mimo że przeciętny gracz może uznać Panzer Battles za zbyt skomplikowany by się nim bawić to jednak entuzjaści gier wojennych i frontu wschodniego będą chyba usatysfakcjonowani.

(Jeff James - Amiga World 8/92)

Szóstka Sierry On-Line
(Sierra On-Line)

Z pewnością wielu z Was pamięta jeszcze gry firmy Sierra On-Line takie jak np. Leisure Suit Larry czy Police Quest. Wczesne wersje gier firmy Sierra On-Line niezależnie od swojej wspaniałej fabuły charakteryzowały się tym, że praktycznie nie wykorzystywały wspaniałych przecież możliwości Amigi (poza tym, że działały w multitaskingu).

Czasy jednak się zmieniają. Pojawiła się właśnie nowa generacja tej serii gier, pozbawiona wcześniejszych błędów i szybsza w działaniu. Dostępne są ulepszone wersje takich gier jak King's Quest V, Leisure Suit Larry, Space Quest IV, Police Quest III, Castle of Dr. Brain oraz nowa wersja oryginalnego Space Quest. Obecne wersje nie działają już w multitaskingu lecz mamy w zamian 32 kolorową grafikę i nowy interfejs użytkownika.

Same gry nie zostały zmienione. Idea pozostała ta sama: należy poruszać się, znajdować i zbierać przedmioty a następnie używać ich w odpowiednim momencie aby przejść dany etap lub odnaleźć inne przedmioty.

W Police Quest III sierżant Sonny Bonds próbuje rozwikłać problem serii morderstw. W Space Quest IV miedzygalaktyczny inżynier Roger Wilco jest przenoszony w przeszłość i przyszłość ścigając Sludge'a Vorhaul'a.

W King Quest V król Graham poszukuje zamku, w którym znajduje się jego rodzina.

Gry wyglądają znacznie lepiej. KQ jest przypuszczalnie najładniejszą grą na Amigę jaką kiedykolwiek zrobiła Sierra. Police Quest III ma bardzo dobre animacje, niestety twarze postaci (mimo 32 kolorów) nie są tak dopracowane.

Niestety wszystkie te gry są nieco wolne na Amigach wyposażonych jedynie w procesor 68000. Wszystkie oprócz KQ V i Larry V mają oznaczenia na opakowaniach - faster Amiga recomended - zalecana szybsza Amiga. Prawdopodobnie także tym dwóm pierwszym by to nie zaszkodziło. Jest to samo w sobie zagadką jak Amiga nie może poradzić sobie z jednym zadaniem.

W SQ IV Wilco pracuje w restauracji fast-foodi pojawia się tam ekwencja w konwencji ar-

cade. Jest to akurat rzecz z którą Amiga nie powinna mieć kłopotów jednak sekwencja ciągnie się niemiłosiernie. Kiedy Bonds prowadzi samochód w Police Quest III, samochód (widziany z perspektywy w małym okienku) jest jedyną rzeczą która porusza się na ekranie - a droga nadal się dłuży.

Czy lepiej się gra z nowym interfejsem? Używanie zbioru ikon jest mniej frustrujące niż zmaganie się z analizatorem wyrazów lecz także mniej wciągające. Wpisywanie może być przykre lecz po pewnym czasie znasz na pamięć każdy cal terenu po którym się poruszasz ponieważ wykorzystałeś każdy dostępny czasownik na wszystkie sposoby i w każdym miejscu. Nowe gry wyraźnie wydają się krótsze niż ich starsze wersje.

Podsumowując, są to gry dla koneserów tego typu zabawy. Z pewnością ci, którzy mieli kontakt z poprzednimi wersjami tych gier z przyjemnością do nich powrócą.

(Peter Olafson Amiga World 8/92)

Computer Third Reich
(Avalon Hills)

Komputerowa Trzecia Rzesza to kolejny przedstawiciel gier "wojennych". C3R jest złożeniem strategii militarnej, ekonomii i elementów polityki pozwalając dwóm graczom odtworzyć przebieg drugiej wojny światowej w Europie. Jeden z graczy kieruje siłami Osi (Niemcy, Włochy i inne państwa) podczas gdy drugi reprezentuje aliantów (Wielka Brytania, Rosja, Francja i USA).

C3R oferuje trzy historyczne scenariusze (odtwarzające położenie sił Osi i aliantów w latach 1939, 1941 i 1944) oraz przeprowadzenie II Wojny Światowej "od nowa".

Aby dopomóc Ci w pokonywaniu trudnej ścieżki do zwycięstwa C3R udostępnia intuicyjny interfejs sterowany myszą. Główny ekran posiada mapę - strategiczne pomniejszenie znacznie większej heksagonalnej mapy Europy po której możesz się płynnie poruszać przy pomocy odpowiednich ikon. C3R wyświetla wszystkie jednostki wojskowe jako prostokąty z wypisaną zdolnością przemieszczania się i siła ofensywną. Ustawianie, poruszanie i atakowanie jednostkami jest proste - wystarczy kliknąć na odpowiedniej jednostce a następnie na celu do którego chcemy się przesunąć (lub zaatakować).

Gra nie montuje się na twardym dysku i wygląda na zabezpieczoną przed kopiowaniem. Pracuje bez kłopotów na wszystkich Amigach wyposażonych w 1MB pamięci RAM. Razem z grą dostarczana jest 19 stronicowa instrukcja (można też zamówić znacznie większą instrukcję: Gamer's Guide to Third Reich).

Brak oszałamiającej grafiki czy muzyki jest doskonale kompensowane przez dobrze wykonany interface z użytkownikiem i przemyślaną koncepcję gry. Jeśli szukasz dobrych gier strategicznych to C3R jest dobrym wyborem!



A2024 - monitor wysokiej rozdzielczości

Niedawno opisywaliśmy w KEBAB'ie Flicker-Fixer "Multivision". Otrzymałem również sporo listów dotyczących tego urządzenia i możliwości jego zakupu.

Niestety w chwili obecnej nie jest to uż możliwe. Co zatem możemy zrobić jeżeli chcemy nie tylko "postrzelać, pozjadać i poudeptywać" na naszej Amidze? Przyjmijmy, że chcemy popracować na przykład używając doskonałego pakietu DTP o nazwie PageStream (najlepiej V2.2).

Mamy standardową Amigę, mamy nawet całe 1MB pamięci i... okazuje się, że nawet jeżeli uda nam się uruchomić nasz program to wyczerpaliśmy już całe zasoby RAM'u. Zatem najpierw kupujemy sobie dodatkową pamięć Fast RAM. Na początek jakieś 4MB.

W połączeniu z 1MB Chip-RAM'u na płycie komputera daje nam to dosyć sensowną (jak na dzień dzisiejszy) pojemność pamięci. Niestety dawno do przeszłości należą czasy, że 64KB RAM'u to było wystarczająco dużo. Dzisiaj nawet niektóre gry wymagają więcej niż 1MB. Jak mamy już pamięć to możemy przystąpić do pracy!

Tylko, że to ciągłe przekładanie dyskietek może doprowadzić do pasji. Pamiętam, że moje pierwsze zetknięcie z PageStream'em spowodowało popłoch wśród wszystkich znajomych posiadaczy zewnętrznych drive'ów do Amigi.

Po pewnych komplikacjach udało mi się w końcu skompletować trzy napędy zewnętrzne co w połączeniu ze sporym RAM-dyskiem dało mi pięć (słownie: pięć) napędów i umożliwiło pracę bez (!) przekładania dyskietek. No! Tylko, że to ciągłe czekanie na załadowanie np. najprostszego fontu dalej doprowadzało do szaleństwa. Tak stałem się posiadaczem twardego dysku. Na szczęście miałem wcześniej kontakt z różnymi napędami typu dhx: i wiedziałem doskonale, że produkt oznaczony symbolem A590 20 (!?) MB to coś co nie tylko nie spełnia oczekiwań użytkownika ale również może doprowadzić go do załamania psychicznego.

Nie mając ochoty odwiedzać w charakterze pacjenta, jednego z kilku w Polsce, szpitali psychiatrycznych, nabyłem napęd Trinology - Quantum LP105. Pierwsze słowo to nazwa kontrolera (sterownika jak kto woli), następne dwa to producent i model samego napędu.

W takim zestawie osiągnąłem pięć razy większą pojemność, cztery razy większą szybkość transmisji, wielokrotnie mniejszy poziom zakłóceń akustycznych w porównaniu do A590 za... 15% większą cenę. Tak! Teraz już miałem wszystko co potrzebne do pracy. Tylko, że to migotanie ekranu w trybie Interlace to właściwość Amigi, która na dłuższą metę uczyni nas częstym klientem okulisty. Jedno rozwiązanie tego problemu, to Flicker - Fixer i monitor VGA lub Multisync. Do zastosowań typu pisanie tekstów czy DTP możemy nawet zrezygnować z kolorów co znacznie obniża koszt takiego zestawu. Przykładowo Flicker-Fixer "Multivision" to ok. 2.700 tys. zł. Monitor VGA-mono to ok. 2.300 tys. zł. W sumie ok. 5 mln zł. na dzień dzisiejszy tj. wrzesień 92.

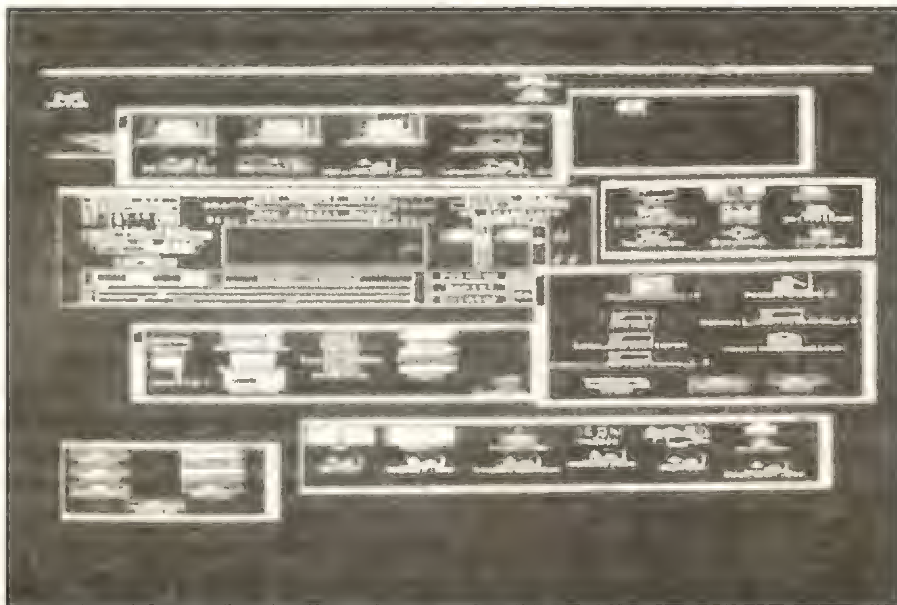
Ale firma Commodore proponuje nieco inne rozwiązanie. Monitor o nazwie A2024. Pełna nazwa tego urządzenia brzmi: Monitor monochromatyczny wysokiej rozdzielczości - A2024. Użytkownicy Workbench'a 2.0 znają zapewne już tą nazwę. Jest to właśnie ten monitor, który możemy wybrać jako podstawowy w nowej wersji systemu operacyjnego (2.0).

Monitor ten posiada 15-o calowy ekran oraz... zintegrowany (tzn. zamontowany wewnątrz obudowy) Flicker-Fixer! Monitor podłącza się bezpośrednio pod gniazdo RGB Amigi. Służy do tego jak zwykle, w przypadku monitorów Commodore, niezbyt długi kabel zamocowany na stałe do monitora.

Po podłączeniu, możemy A2024 wykorzystywać jako zwykły monitor monochromatyczny. Niestety wykorzystuje on sygnały cyfrowe RGB Amigi i pozwala na uzyskanie maksymalnie czterech kolorów, a właściwie czterech odcieni szarości. W przypadku korzystania z Workbench'a jest to w zupełności wystarczające.

Flicker-Fixer działa bez zastrzeżeń. Interlace nie migocze a "normalny" tryb 256 linii jest tak jak





w przypadku innych FF-ów odtwarzany bez denerwujących prążków międzyliniowych. Testowany egzemplarz posiadał niestety niezbyt dobrze wyregulowaną geometrię obrazu.

Powodowało to swoiste zaokrąglenia linii prostych w rogach ekranu. Jak wynika z rozmów z dostawcą jest to dosyć częsty objaw. Dlatego uprzedzamy! Przy zakupie należy dokładnie sprawdzić i ewentualnie wybrać najlepszy egzemplarz. Zresztą dotyczy to również pozostałych modeli monitorów firmy Commodore.

Ale wróćmy do tematu! A2024 to nie tylko zwykły monitor monochromatyczny razem z flicker-fixerem. To również, a może przede wszystkim zupełnie nowa jakość (?!).

Nowe tryby rozdzielczości. Oprócz standardowych 320x256 i 640x512 mamy do dyspozycji tryb o nazwie takiej jak monitor (A2024) i rozdzielczości 1008x1024... oczywiście bez "mrugania". Do czego może nam się to przydać?

Workbench może mieć cztery razy większe wymiary, czyli możemy otworzyć więcej okien, większe okna itd. Co prawda na zwykłej, nie przyspieszonej Amidze praca z takim dużym Workbench'em jest dosyć trudna. Tym bardziej, że wymaga min. 1MB Chip-RAM'u. Przydać się natomiast taka rozdzielczość może w przypadku pracy z niektórymi programami np. do DTP gdzie wyższe rozdzielczości są przydatne.

Kto zatem powinien kupić sobie taki monitor? Każdy kto ma zamiar spędzać dużo czasu przed ekranem Amigi! Każdy, kto ma zamiar pisać teksty, programować (w sumie też pisać teksty), pracować z programami DTP etc. A2024 stanowi najtańsze rozwiązanie problemu "migotania" oraz daje nam dodatkowe możliwości.

SD!

Producent: CBM

Produkt: Monitor A2024

Dostawca: Silver Dream !'s

Cena (9/92): ok. 4mln zł.

\+ tryby wysokiej rozdzielczości

\+ wysoka ostrość obrazu

\+ zintegrowany flicker-fixer

\+ stosunkowo niska cena

\- nienajlepsza jakość wykonania

\- obsługa tylko czterech odcieni szarości

\- krótki kabel przyłączeniowy

Ocena KEBAB'a (1 do 6): 4 - dobra

OGŁOSZENIA DROBNE

Kupię używaną Amigę 500
za 5,6mln na 2 raty.
1. 3.000.000,-
2. 2.600.000,-
Rafał Piskow
Murzynowskiego 10/175
10-684 Olsztyn

THE GROUP AGNUS! (C-64)
nawiąże kontakt z uczącymi
się Assemblera. UWAGA!
Poszukujemy: grafików,muzy-
ków, swaperów, coderów!
Rafał Tylus
ul. Niepodległości 16/36
62-400 Słupca

Korespondencyjny kurs
AMOSA dla początkujących.
Informacje:
Bogdan Ożorowski
ul. Słowackiego 58/13
41-219 Sosnowiec

Sprzedam gry oraz programy
użytkowe na C-64 (Informacje:
koperta+znaczek) . (kasety)
Mariusz Listowski
ul. Sobieskiego 17/8
76-200 Słupsk

Sprzedam dysk twardy -
QUANTUM 52 MB
do Amigi 500
Tomasz Chojnacki
ul. Pow. Śląskich 57
32 -400 Olkusz
tel. (0 - 35) 431571

Sprzedam książkę
"MC 668000" J. Kostrzwskiego
cena 100.000zł
(za pobraniem pocztowym)
Piotr Łaszczyk
34 - 511 Kościelisko 976

Pilnie poszukuję muzyka i
grafika do współpracy.
MR SOFT, tel. (091) 820992

Sprzedam cartridge Quick Box
(asembler, program syntezy
polskiej mowy i dzwięku,
Black Box 81 do C-64. Cena
do uzgodnienia
.Mączak Mariusz
Os. Brzozowe 4/29
11 - 700 Mrągowo

Pilnie sprzedam C-64, mag-
netofon, stację dysków 5,25"
1541 II, 35 dysków z na-
jnowszymi grami i programami,
Final III, joystick.
Marcin Rudnicki
ul. Chełmińska 8
82 - 500 Kwidzyń
tel. 27 -13

Amiga - PROGRAMY
(koperta + znaczek)
Henryk Bagiński
B. Głowackiego 2
68-200 Żary

Sprzedam gry i programy na
C-64. Cena dysku - 15.000 zł,
kaseta - 20.000 zł. Spis gratis!
Wojciech Tomaszewski
ul. Strażacka 7
68-205 Żary

Odsprzedam mapę pamięci
C-64 po polsku - 140.000,- zł.
Krzysztof Sutkowski
Kinkajmy 15/11
11-200 Bartoszyce

Sprzedam literaturę, oprogra-
mowanie i osprzęt (np. pióro
świetlne) do C-64.
Tomek Kępa
ul. Stawisińskiego 4Ł/68
86-100 Toruń
tel. 488 - 523

Amiga: wymienię programy,
nawiążę kontakt z grającymi
w EYE OF THE BEHOLDER II
Artur Święch
ul. Złota 20/3
25-015 Kielce

Sprzedam C-64, magnetofon,
gwarancja, joystick, mysz,
pokrywa, literatura.
Cena 2-2,5 mln.
Krzysztof Wawrzyniak
ul. Wyszyńskiego 6/16
62-510 Konin
tel. 426 - 530

Sprzedam obrazki do ART -
STUDIA. 20 obrazków + dysk
25.000,- zł.
Jacek Szumowski
ul. Sikorskiego 6A/40
75-360 Koszalin

Ponad 3.000 tytułów progra-
mów na Amigę wymienię lub
sprzedam.
Wojciech Knapczyk
ul. Gotarda 1/36
02-683 Warszawa

Zamienię C-64 II z magneto-
fonem, 2 joystickami, myszą,
2 modułami, 300 programów
na używaną Amigę z ew. do-
płatą do 1 mln.
Piotr Duszyński
ul. Startowa 7C/26
Gdańsk - Zaspa

Public Domain C-64, dyskietki
i kasety. Wyjątkowa okazja!
Bardzo tanio.
Koperta + znaczek
Marcin Lis
ul. Kochanowskiego 14B\20
01-864 Warszawa

Sprzedam A500 z 1MB+
monitor 10845 + filtr + sampler
+ oryginalne programy
Bartosz Machoński
ul. Majowa 2/2
Szczecin tel. 526 - 321

Gry, użytki - C-64, Katalog
- 10.000 zł, (na kasecie)
Tomasz Krupnik
Janów 87
96-512 Młodzieszyn

Sprzedam C-64, magnetofon,
Black Box 3, 10 kaset, litera-
tura (1,6 mln), mysz 250.000 zł.
M. Mocek
ul. Daszyńskiego 7/2
88-100 Inowrocław

TSM - podaj adres MR.RAD
Radosław Twardzik
Przemysłowa 31
22-100 Chełm

THE GROUP AGNUS (C-64)
Attention...Attention...Attention...
Poszukujemy (very!) muzyków
i swapperów !!! Nawiążemy
kontakty! Grupy i grupki
odezwijcie się!!!
Rafał Tylus
ul. Niepodległości 16/36
62-400 Słupca

Poszukujemy swaperów
chętnych do współpracy
z naszą grupą. Piszcie!
Hunter of Perocity
ul. 1-go Maja 17/19 m 3
88-200 Rodziejów

Sprzedam gry i programy na
C-64. Cena dysku 15.000,- zł,
kaseta - 20.000,- zł.
Spis gratis!
Wojciech Tomaszewski
ul. Strażacka 7
68-205 Żary

Sprzedam Amigę 500 (v: 1.3.2,
1 MB), monitor,literatura,
60 dysków. Cena ok. 9 mln.
Piotr Kołodziejczyk
ul. Ludowa 47/1
58-304 Wałbrzych

Sprzedam C-64 w dobrym
stanie + osprzęt.
Kontakt listowny:
Filip Tycman
ul. Pomorska 4B/6
89-300 Wyrzysk

Sprzedam: A500 1 MB+monitor
mono + stacja 5,25" + 150 dys
ków + wiele dodatków - 11 mln
z LC 20 - 14,5 mln.
Piotr Mikulski
Adamieckiego 11/115
41-300 Dąbrowa Górnicza

Kupię stacje 1541 II
Oferty z ceną:
Krzysztof Najborowski
Os. Batorego 4/56
60-687 Poznań

Odstąpię wiele ciekawych gier
i programów na C-64.
Informacje/zamówienia:
A. Makowski
Box 419
45-076 Opole 1

Sprzedam: C-64, magnetofon,
joysticki, Black Box 4 oraz
300 programów. Cena 1,7 mln.
Wojciech Chałat
ul. Aleja 8/44
42-200 Częstochowa

Przyklad zastosowania przerwan
Zegar
MR SOFT - Kebab '92

Z powodu ogromnych kłopotów organizacyjnych podczas składu poprzedniego numeru Kebaba, zaginął listing do artykułu "O przerwaniach", który to zamieszczamy poniżej, jednocześnie gorąco przepraszając Czytelników i autora.

Redakcja.

```
Clock:      movem.l   d0-d7/a0-a6,-(sp)

            move.w    $dff01e,d0
            and.w     #$0020,d0                  ;czy przerwanie VBLK ?
            bne       AddTick
            bra       Out1

Out:        move.w    #$0020,$dff09c
Out1:       movem.l   (sp)+,d0-d7/a0-a6
Next:       jmp       0.1                        ;skok do poprzedniej procedury


Addtick:    addq.b    #1,Tick
            cmpi.b    #50,Tick                   ;50 tykniec na sekunde (NTSC-60)
            bne       Out

            move.b    #0,Tick
            addq.b    #1,Sec
            cmpi.b    #60,Sec
            bne       Out

            move.b    #0,Sec
            Addq.b    #1,Min
            cmpi.b    #60,Min
            bne       Out

            move.b    #0,Min
            addq.b    #1,Hour
            cmpi.b    #24,Hour
            bne       Out

            move.l    #0,Hour                    ;przepelnienie
            bra       Out


Install:    move.w    $dff01c,Buffer
            move.w    #$7fff,$dff09a             ;wylaczenie przerwan
            move.l    $6c,OldVect                ;Zachowaj poprzedni wektor
            move.l    #Clock,$6c

            lea       Next(pc),a0
            move.l    OldVect(pc),$02(a0)
            move.l    #0,Hour                    ;zerowanie zegara
            move.w    Buffer(pc),d0
            or.w      #$8020,d0                  ;ustawienie bitow SET/CLR i VBLK
            move.w    d0,$dff09a                 ;wlaczenie przerwan
            moveq     #0,d0
            rts


Remove:     move.w    #$7fff,$dff09a
            move.l    Oldvect,$6c                ;Przywroc stary wektor
            move.w    Buffer,d0
            or.w      #$8000,d0
            move.w    d0,$dff09a
            moveq     #0,d0
            rts

OldVect:    dc.l      0
Buffer:     dc.w      0
Hour:       dc.b      0          ;Godziny
Min:        dc.b      0          ;Minuty
Sec:        dc.b      0          ;Sekundy
Tick:       dc.b      0          ;Tykniecia
```

Listing nr 1

Tworzenie biblioteki bootblockow
dla programu BootX
by Thorgal/W.F.M.H.
(c) 1992 Kebab

```
Start
      dc.l  'BXBB'        ; identyfikator
      dc.l  420           ; numer wersji
      dc.l  number        ; ilosc bootblockow w biblio-
                          ; tece
      dc.l  number        ; wartosc aktualnie nieuzy
                          ; wana - ustawic jak wyzej
      dc.l  0             ; ktory bootblock ma byc
                          ; pokazany jako pierwszy
                          ; (UWAGA: numeracja od
                          ; zera!)

      incdir              "ram:"

number      set           0

Boot        macro
name\@      dc.b          \1

      if    *-name\@>>30
      fail
      endc

      blk.b 30-(*-name\@),0
      incbin              \2

      if1
number      set           number+1
      endc

      endm

; Ponizej przyklady uzycia makrodefinicji Boot.
; Usun je i wprowadz wlasne.

   Boot  "AIDS Protector II","AidsPtII"
   Boot  "Dos Installed","normal"
```

Listing nr 2

```
auto rs\$60000\0\2\
auto wb\$60000\$60400\
```

Listing nr 3

Przelaczenie na tryb NTSC
dla "starego" Agnusa
by Thorgal/W.F.M.H.
(c) 1992 Kebab

```
synchro EQU        $10400
   lea             $DFF000,a4
CheckRaster
   move.l          4(a4),d0
   and.l           #$1FF00,d0
   cmp.l           #Synchro,do
   bne.s           CheckRaster
   move.l          4(a4),d1
   and.l           #$80000000,d1
   or.l            #$13778,d1
   move.l          d1,$2A(a4)
   btst            #6,$bfe001
   bne.s           CheckRaster
   rts
```

Listing nr 4

Programowe przelaczenie
na tryb NTSC.
by Thorgal/W.F.M.H.
(c) 1992 Kebab

```
CoolCapture   =   46

   move.l          4.w,a6
   lea             ResetProc(pc),a1
   move.l          a1,CoolCapture(a6)
   bsr.s           SumLibrary
   rts

ResetProc:
   move.w          #0,$dff1dc        ; bclr w tej
                                     ; sytuacji
                                     ; zawodzi
   clr.l           CoolCapture(a6)

SumLibrary:
   lea             34(a6),a1
   clr.w           d0
   moveq           #24-1,d1
loop:
   add.w           (a1)+,d0
   dbf             d1,loop
   not.w           d0
   move.w          d0,(a1)
   rts
```

Listing nr 5

Set Reset Vector
by Marcin 'MR SOFT' Or+owski

```
AllocMem  equu  -198

Begin:
   move.l      4,a6
   move.l      #$10002,d1
   move.l      #[RoutEnd-Routine]+1,d0
   jsr         AllocMem(a6)
   move.l      4,a6
   move.l      d0,$2a(a6)          ;$2A = 42dec

   lea         Routine,a0
   move.l      d0,a1
   move.l      #[RoutEnd-Routine],d0

Copy:
   move.b      (a0)+,(a1)+
   subq.l      #1,d0
   bne         Copy

   lea         $22(a6),a1          ;$22 = 34dec
   move.w      #$17,d1
   clr.l       d0

CheckSum:
   add.w       (a1)+,d0
   dbf         d1,CheckSum
   not.w       d0
   move.w      d0,(a1)
   rts

Routine:
   sub.w       d0,d0

ResLoop:
   move.w      d0,$dff180
   addq.w      #1,d0
   andi.w      #$fff,d0
   bra         ResLoop
RoutEnd:
```

Drodzy Czytelnicy!

Aby umożliwić Wam również wpisywanie dłuższych programów np. takich jak prezentowana w niniejszym numerze gra: Mini - Penetrator lub dłuższych, postanowiliśmy wprowadzić jeszcze jeden program do wpisywania takich właśnie listingów.

O ile wpisywanie przy użyciu Korektora lub monitora pamięci było o tyle wygodne, że wpisywaliśmy dokładne wartości heksadecymalne o tyle przy użyciu Inputera zaoszczędzamy sporo procent pracy (uderzeń w klawisze) gdyż program jest odpowiednio zakodowany aby skrócić do minimum czas wpisywania.

Dekodowanie programu odbywa się automatycznie po wprowadzeniu. Oczywiście nie oznacza to, że zrezygnujemy całkowicie z drukowania listingów w formacie Korektora. Wszystkie krótsze listingi będzie można dalej wpisywać "po staremu".

```
0 REM * CODE-INPUTER (C)1992 KEBAB
1 REM * BY PAWEL SOLTYSINSKI
2 REM * LISTING MADE WITH CODELISTER V3
3 REM * ----------------
4 :
5 PRINT:PRINTCHR$(5);"PROSZE POCZEKAC 1 MIN.":PRINT
10 FOR T=20480 TO 21688:READ A$
20 GOSUB 50:POKE T,A:B=B+A:NEXT T
30 IF B<<>>124532 THEN PRINT "ZLE DANE!":STOP
31 PRINT"(T)ASMA CZY (D)YSK ?":X=8
32 GETA$:IF A$<<>>"T" AND A$<<>>"D" THEN 32
33 IF A$="T" THEN X=1
34 POKE 20483,X
40 SYS 20480:REM START PROGRAMU
50 B$=LEFT$(A$,1):GOSUB 70:A=C*16
60 B$=RIGHT$(A$,1):GOSUB 70:A=A+C:RETURN
70 C=ASC(B$):IF C>>64 THEN C=C-55:RETURN
80 C=C-48:RETURN
90 :
1000 DATA 4C,A9,51,01,30,31,32,33,34,35
1001 DATA 36,37,38,39,01,02,03,04,05,06
1002 DATA 07,08,09,0A,0B,0C,0D,0E,26,10
1003 DATA 11,12,13,14,15,16,17,18,19,1A
1004 DATA 41,42,43,44,45,46,47,48,49,4A
1005 DATA 4B,4C,4D,4E,21,50,51,52,53,54
1006 DATA 55,56,57,58,59,5A,23,25,05,C3
1007 DATA 4F,44,45,50,41,43,4B,20,34,2F
1008 DATA 33,20,C9,4E,50,55,54,45,52,20
1009 DATA 42,59,0D,00,D0,4F,4C,4F,4E,55
1010 DATA 53,2F,D0,41,44,55,41,2F,CB,45
1011 DATA 42,41,42,0D,0D,00,53,54,41,52
1012 DATA 54,20,41,44,44,52,45,53,53,20
1013 DATA 28,46,31,29,3A,20,24,30,38,30
1014 DATA 31,0D,00,43,55,52,52,45,4E,54
1015 DATA 20,45,4E,44,20,4F,46,20,50,52
1016 DATA 47,3A,20,24,30,38,30,31,0D,0D
1017 DATA 00,46,33,20,2D,20,4C,4F,41,44
1018 DATA 0D,00,46,35,20,2D,20,53,41,56
1019 DATA 45,0D,0D,00,3A,20,30,30,30,30
1020 DATA 20,30,30,30,30,20,30,30,30,30
1021 DATA 20,30,30,30,30,20,30,30,30,30
1022 DATA 20,30,30,30,30,20,30,30,20,20
1023 DATA 06,07,08,09,0B,0C,0D,0E,10,11
1024 DATA 12,13,15,16,17,18,1A,1B,1C,1D
1025 DATA 1F,20,21,22,24,25,00,AE,00,51
1026 DATA BC,E6,50,B9,98,07,49,80,99,98
1027 DATA 07,60,48,4A,4A,4A,4A,20,1C,51
1028 DATA A8,68,29,0F,C9,0A,B0,04,09,30
1029 DATA AA,60,E9,09,AA,60,A9,00,85,02
1030 DATA 98,20,37,51,0A,0A,0A,0A,85,02
1031 DATA 8A,C9,30,B0,04,69,09,D0,02,E9
1032 DATA 30,05,02,85,02,60,A9,00,85,20
1033 DATA AA,BD,95,04,49,80,9D,95,04,20
1034 DATA E4,FF,F0,FB,C9,0D,F0,1E,C9,30
1035 DATA 90,F3,C9,47,B0,EF,C9,41,B0,04
1036 DATA C9,3A,B0,E7,29,3F,A6,20,9D,95
1037 DATA 04,E8,8A,29,03,4C,48,51,A6,20
1038 DATA BD,95,04,49,80,9D,95,04,AC,95
1039 DATA 04,AE,96,04,20,28,51,8D,A6,51
1040 DATA 8D,A8,51,AC,97,04,AE,98,04,20
1041 DATA 28,51,8D,A5,51,8D,A7,51,4C,A9
1042 DATA 51,01,08,01,08,20,18,E5,A9,0E
1043 DATA 20,D2,FF,A9,08,20,D2,FF,A9,80
1044 DATA 85,9D,A9,09,85,D3,A9,44,A0,50
```

```
1045 DATA 20,1E,AB,A9,0B,85,D3,A9,5E,A0
1046 DATA 50,20,1E,AB,A9,08,85,D3,A9,74
1047 DATA A0,50,20,1E,AB,A9,08,85,D3,A9
1048 DATA 8F,A0,50,20,1E,AB,A9,0F,85,D3
1049 DATA A9,AB,A0,50,20,1E,AB,A9,0F,85
1050 DATA D3,A9,B6,A0,50,20,1E,AB,A0,27
1051 DATA A9,2D,20,D2,FF,88,10,FA,AD,A6
1052 DATA 51,20,10,51,8C,95,04,8E,96,04
1053 DATA AD,A5,51,20,10,51,8C,97,04,8E
1054 DATA 98,04,A2,04,BD,BE,50,9D,98,07
1055 DATA E8,E0,28,90,F5,CA,A9,01,9D,98
1056 DATA DB,CA,10,FA,AD,A8,51,20,10,51
1057 DATA 8C,98,07,8C,BD,04,8E,99,07,8E
1058 DATA BE,04,AD,A7,51,20,10,51,8C,9A
1059 DATA 07,8C,BF,04,8E,9B,07,8E,C0,04
1060 DATA A9,00,8D,00,51,20,01,51,20,E4
1061 DATA FF,F0,FB,C9,85,D0,03,4C,46,51
1062 DATA C9,0D,D0,03,20,B2,53,C9,86,D0
1063 DATA 03,4C,25,54,C9,87,D0,03,4C,8F
1064 DATA 54,C9,1D,D0,0F,20,01,51,EE,00
1065 DATA 51,AD,00,51,C9,1A,B0,C6,90,C9
1066 DATA C9,9D,D0,0C,20,01,51,CE,00,51
1067 DATA 10,BD,A9,19,D0,B6,20,B9,52,B0
1068 DATA B7,48,20,01,51,68,99,98,07,20
1069 DATA DB,52,4C,88,52,38,60,C9,21,90
1070 DATA FA,C9,DB,B0,F6,C9,60,B0,05,29
1071 DATA 3F,4C,CC,52,29,5F,A2,3F,DD,04
1072 DATA 50,F0,05,CA,10,F8,30,DF,18,60
1073 DATA 00,AD,A7,51,18,6D,A8,51,85,FD
1074 DATA A9,00,8D,DA,52,A2,00,BC,E6,50
1075 DATA B9,98,07,8D,A0,01,B9,99,07,8D
1076 DATA A1,01,B9,9A,07,8D,A2,01,B9,9B
1077 DATA 07,8D,A3,01,8E,2C,53,A2,03,BD
1078 DATA A0,01,A0,3F,D9,04,50,F0,03,88
1079 DATA 10,F8,98,A0,05,4A,6E,08,01,6E
1080 DATA 09,01,6E,0A,01,88,10,F3,CA,10
1081 DATA E0,A9,00,18,69,04,48,A0,00,AE
1082 DATA DA,52,B9,08,01,9D,B8,02,EE,DA
1083 DATA 52,C8,C0,03,90,EF,68,AA,E0,18
1084 DATA 90,A1,AD,BC,07,48,AD,BD,07,AA
1085 DATA 68,A8,20,28,51,85,FE,A0,00,B9
1086 DATA B8,02,8D,64,53,98,AA,A9,00,18
1087 DATA 65,FD,85,FD,CA,10,F6,C8,C0,12
1088 DATA 90,E9,A5,FD,C5,FE,D0,03,A9,2B
1089 DATA 2C,A9,20,8D,BF,07,60,A9,90,85
1090 DATA 20,A9,B8,85,22,A9,05,85,21,85
1091 DATA 23,A2,0D,A0,27,B1,22,91,20,88
1092 DATA 10,F9,A5,20,18,69,28,85,20,90
1093 DATA 02,E6,21,18,69,28,85,22,90,02
1094 DATA E6,23,CA,D0,E0,60,AE,BF,07,E0
1095 DATA 2B,D0,F8,AD,A7,51,85,FB,AD,A8
1096 DATA 51,85,FC,A0,00,B9,B8,02,91,FB
1097 DATA C8,C0,12,90,F6,98,18,6D,A7,51
1098 DATA 8D,A7,51,90,03,EE,A8,51,20,7B
1099 DATA 53,20,01,51,20,81,53,68,68,4C
1100 DATA 1E,52,93,0D,0D,C5,4E,54,45,52
1101 DATA 20,46,49,4C,45,4E,41,4D,45,3A
1102 DATA 00,A9,EA,A0,53,20,1E,AB,A2,00
1103 DATA 20,CF,FF,C9,0D,F0,08,9D,B8,02
1104 DATA E8,E0,10,90,F1,8A,A2,B8,A0,02
1105 DATA 20,BD,FF,A9,0D,20,D2,FF,4C,D2
1106 DATA FF,20,FD,53,AD,03,50,C9,08,D0
1107 DATA 46,AA,A9,01,A0,00,20,BA,FF,20
1108 DATA C0,FF,A2,01,20,C6,FF,20,CF,FF
1109 DATA 85,FB,8D,A5,51,20,CF,FF,85,FC
1110 DATA 8D,A6,51,20,CF,FF,A0,00,91,FB
1111 DATA E6,FB,D0,02,E6,FC,A5,90,F0,EF
1112 DATA A9,01,20,C3,FF,20,CC,FF,A5,FB
1113 DATA 8D,A7,51,A5,FC,8D,A8,51,4C,A9
1114 DATA 51,A2,01,A9,00,A8,20,BA,FF,A9
1115 DATA 00,20,D5,FF,A2,03,BD,3D,03,9D
1116 DATA A5,51,CA,10,F7,30,E3,20,FD,53
1117 DATA AE,03,50,E0,08,F0,02,A2,01,A9
1118 DATA 01,A8,20,BA,FF,AD,A5,51,85,FB
1119 DATA AD,A6,51,85,FC,A9,FB,AE,A7,51
1120 DATA AC,A8,51,20,D8,FF,4C,A9,51

READY.
```



"MINI-PENETRATOR"

```
$0801-$1988
0801: 2Myg 1FU! c3gV 0a00 ue&1 KtA& 63
0813: CvE0 !d3T j001 1MYe f%L% Se3M 70
0825: sfNe qL%X fMUf 1Nnr XL%v %73M 4e
0837: U820 15aY fDMZ iy0& 92jS 1mxq 8e
0849: Cqnr CnX3 SYfv T%%u 8KV% v#XE 6d
085b: 1WVu U&7G jAFa al3c XDvK Pdk7 7e
086d: &7xM &fg& pDp# rC&l ZDX0 gDV0 2b
087f: d90w h3M6 300& lgws 7ykC azE% 6f
0891: KeP1 FAAD ggQ0 rpfK uHKf uBqI 80
08a3: uyCk rIsG kr7j 0u&P bl0k %uyZ 9b
08b5: qw5B cwEJ y4W0 GyeA p9Y! vpwl f3
08c7: 1LGU %KfN LHTh KwIZ 0jQ1 Wj9m 6a
08d9: Bb1P 150h 542h 9dq4 Gk1m GB06 le
08eb: HF0m %XD7 %Ziy F1L% B1GM uxmW 6f
08fd: A4mF Hk&F 11mb D5jP eYs4 411g 53
090f: lqAG lLB0 5KBg 6u#g mGts HH#g 1a
0921: 6SWg 6LFg 6ZZ0 mZ0H Ku1z qktl 25
0933: 01gl BAp0 elKG Mz62 050g z!k6 df
0945: hhgP AlY5 6h1C rA01 mg06 h00r 73
0957: Bg0m gk05 !Bs& 17%d cNE4 40sx f0
0969: 5g6A 005A GNlg 10qg 6k5g 6A10 04
097b: 5hgB v0J0 du1% eE2E Wg1c 4x1c 36
098d: nXlZ XQ51 w810 %UyB %G0f dDds 9f
099f: lefH CncH CtP3 1iNU 4dip 402p 99
09b1: lyS8 4dTD 4#7m &853 %I62 5Rph 69
09c3: 9!vl PUra ycGU 6Y#2 %M61 4v%f c2
09d5: 9CIu NxYg 1GGg xh%6 7G82 FhUg 8e
09e7: 4HRT atQD kwmF %B99 TLDv BbRZ 81
09f9: hfdm Chgi wxd2 Q3qZ %rqp 0b48 3d
0a0b: SVA1 m9ip 0SsI EFA2 RfhI lPTS 10
0a1d: Cgjk !x2R &bwg vGr# LFnk 23QY dd
0a2f: atF4 xqL5 67tW J1aV KPiU XUwg 46
0a41: KqEl kM#Q %Ra9 sdUC lJ0X LsQ& c2
0a53: thza 370b g0yY e3ak 64k6 &gBT 43
0a65: sz38 Kicj c3Q7 NYtw 4f61 Ct4U 02
0a77: %uUk Bhz# 3&FS apnf Zq&# s0Cn 3a
0a89: 3&Rz j3r2 pHt5 b88j p2Mw y6ir 34
0a9b: 8%%x F0ue UKF1 Mb0D ep0c yhSt 1b
0aad: fHku wb24 zS2g hDkv bMs5 xpM7 ab
0abf: vvyu ddu0 !9gl ZxJn bz&W !CrJ 81
0ad1: grCm 5kwN 2AyV rCMK Qhep 1Gmp 06
0ae3: 1thE gMFa Tg0z I0nN Q0&0 kKPh d7
0af5: iv#t ciwI T#By s!Cf Y1Eg 21qH 01
0b07: H347 Die0 LurN y5C& VG4& Yc&K b5
0b19: LhAm DgEm j9cg QL&f pN04 4cHd 60
0b2b: PIcw %Xkr Y4An 5NAM dwzM cGyF 59
0b3d: 0FQJ JHAe nLN6 426F #Hhb Q1Rg 2f
0b4f: SX0N 90z7 1LLF nGjr 2ft7 5NIy 6b
0b61: f6t% z&m6 sj!8 !EtF A4Mg b2Af 2f
0b73: i%SN 58kg Gh25 PpWb G4DH 5Rwq a2
0b85: 2#ty 48Tu 4Gz8 Ih2Q 7T3! &W68 e6
0b97: Y1ru !h6d Fgcw 4&aY 43Vm MJRs 6d
0ba9: PbTF 5qwF 1HEe Lsuy Fi2t 11pQ a8
0bbb: %wsm C2D0 Y4GY W7&F 2L%g 4rMj d1
0bcd: 5HB5 4I07 A7BN cg4T Jhj9 %L0H 81
0bdf: zvEi Tkat XNmZ #TNB 1rTR 5t0p 95
0bf1: uKPJ 6!B% 40g& vm8N GbDl 4VT# ae
0c03: 5rA% 59Q1 5C2Y 5xrg XKQX 8dF8 89
0c15: GfZz 47Y7 &MSP gNA5 eJae q0ec 75
0c27: JCa0 0UT2 8vZH 0&9b fNQQ 0N8c 5b
0c39: 4gMj qFK2 &MC0 pmKR AiQ6 3mwb 91
0c4b: W9v% w7Y3 &wg8 &BMH d024 98wf 8f
0c5d: &x27 j85U 6U1C UHe1 &MX% QMcG fc
0c6f: nmW9 9jHa 8%v6 wzri %%xw 5!7w 26
0c81: ejt1 gg15 00ue M&z% 8Re6 K%gM 94
0c93: sbjX hVzJ hWsc t#Bx UmzT zP3q de
0ca5: zQU& XZb3 MZ7L 7S2R 7FMN TWkd 5e
0cb7: XSKy TPX1 qPMV &XVb 3MN5 L#zU 3e
0cc9: ICxC 8we2 3zQk aUGl e0xa 2U8a 76
0cdb: !0L% !5A0 0Mc4 60kU EM47 ZMA9 20
0ced: 2wIb 30Qe 3wYg 4h8a TSIm 5NAq b5
0cff: 71Qv 8icB 9!EI bP8R e3I% gQvz d4
0d11: Z%Ab 7ZFM VgG4 0YU7 N%s8 wEgU f2
0d23: zYu6 x0k2 ZiEk Pmkz cdaF VpGH 2b
0d35: pdtg x9pA 7vY6 3A1r tlrj 10s0 dc
0d47: 7405 Mpb1 E3w3 2fY0 10D% P0ah 9a
0d59: 8Y!v DpK2 wFH% 8&60 &E12 wU8x ab
0d6b: gC8x 8m9y 03rK 5%6r BVih zE!b 62
0d7d: yEC9 y8y7 xUq6 x&nH qUf# yvkw c7
0d8f: 7wzX Nch5 xci2 MEi2 Nf#2 61sm fa
0da1: 518g 3wMb 2wA8 1M&5 10dy vXA1 cd
0db3: 0fW2 x0&c VV7% LGwb 1Acz iSw8 dd
0dc5: 2cy2 M8g6 xdQd 2McT %UDG W6y8 e3
0dd7: !9y9 m6wU ylAE 602t bW8A 8yc! bc
0de9: F2v2 xIe2 pDbI Iyr% %NAE Y%ep 0f
0dfb: 1L0E 89XM NEwI %Ngk M6Ml w&Yl 80
0e0d: 5h4h 5h3& 51cj 5c62 !m0m kw2j 68
0e1f: 2I75 &hVw 8cD5 x6gy &!3% !U88 1d
0e31: hdYw !YPb 1y4d %%W1 8&cf &Eli 8f
0e43: 16a6 0gch k00h gh5j hU&3 1id0 55
0e55: &43M 3gwe 0xMh 0gw6 204a 0gII e3
0e67: 8088 4HU0 07z& ELKq Gt25 %bzn 32
0e79: #W0P x06E nqC% %G80 CLXX M07M 28
0e8b: 2c05 !0bh wgf1 rKQR fYqd xMew cc
```

```
0e9d: 1X4# R%TF BzGB #!x5 #UnZ WhfY 6d
0eaf: VLWB %IAa AcaF dUk1 Ewqe pXWK b3
0ec1: MGKF 02aZ E2tK 4wEC 0Uk2 Fgdk 50
0ed3: tAW5 0UGj 2GF0 1bRw 2F42 Wcz0 25
0ee5: 293R PwYl 4cRc bhG5 H8iJ GB#E f9
0ef7: Hf09 8db% 8dLY jbFd ZNY0 bmAu 88
0f09: jvEI jM#F bqC0 pdL6 Gv4S e1jD f5
0f1b: HLgh dqs4 PIwm &1RI Fhj3 !jGg 49
0f2d: 0GAM M1Hd A1Aq jG85 Lh01 D&s4 2b
0f3f: SIkV &0w5 AFcl waBh 7bxl 3hW7 48
0f51: X!J7 hBw! mJnc lQkB 82bn d2aP 06
0f63: 5QYB zgtf N!pj ef7& xkRl MT1N 82
0f75: 6XNb myx9 CFF1 kBg9 wda2 pcIV c0
0f87: mngd 7exg kAZ7 BdVH S5iX 84#b eb
0f99: FAa# I4Jf l#u9 85u2 HgFB mBBz 5b
0fab: Bkm% 3pMA waI0 f5#2 FszH 3z5k aa
0fbd: tXAK E1w6 6pc5 A8Sq jA5a j4lg 85
0fcf: &qtB 14Z3 imQq 9qyr 5kJM bkV8 f1
0fe1: ctNa k4Zc IY&U 3kkT gK&! 1Bd9 45
0ff3: j5p5 mFB4 %rB1 jp7j R4P8 cZN2 f7
1005: ed82 Sjwm czgT kkLM EG!5 8SgR 82
1017: rGk8 lk5b gkkx 6i5P dyKR jQVp 46
1029: jph2 dKC8 Q3uU Bkt1 jAji dA&f e4
103b: Qjcm czxT ckRj c5IM Kbki 8z0V bf
104d: xQF2 kBlj i2JA 8Pls 2eYq Sz4M 2b
105f: bD91 mk9F kBBf li5b 6kg! u3&0 77
1071: fFlD d3Uw jlwW !!Rg hkV5 15b9 0a
1083: fGIw f0Ss a0lw C4cF cjAV cy12 93
1095: mi13 jQRd jQhf kHW# BQJ5 gA52 cd
10a7: n0Qd DBJ8 ul11 jAJk lynB 033k 2b
10b9: 03hk glhb ijEw cMQw k45c iltf 8a
10cb: exNY vhe& SU1R nwl8 9ivg Y0b6 42
10dd: N59v 3aCz 2NnZ 81zB qPAs GiQs 55
10ef: 3Akq DcE3 5N57 zi7g xsdv EN1U 5a
1101: 777x m4!d 6d2F 4G0n P6re 8pSF ff
1113: 125Y ymc1 XY8N krHM 1Yr3 QeRc 62
1125: &xFc 3!2Y g8Sq 8g8c MuKM s5Mg a9
1137: Pdz3 pcBu h0yG 7Eg4 hDKT !igb 97
1149: 6ywy Hv#h Ngzg 3WWQ W7ye R7cS 0b
115b: x5W1 300s fLVA zgqd eH6F WfyF ee
116d: JM67 Mu19 2cPg 81Hv XDg2 Q0WF fc
117f: U4qq LGSV 28Q5 2t!A btGw igMK f1
1191: KWjK zSU! KS6W M90g Gq2d ayAH 13
11a3: WkTm 5CB8 R!!Z KN7& GcKK YB6L 80
11b5: HE9c rscA TEpp WGyL fGWJ 0yl1 9e
11c7: irXc j8aJ a&em HCS9 MWmL e82J 85
11d9: qipQ HLwa HKAa HPDv j6sr E05q 60
11eb: r8uK RArg L#02 I1BU 6xiu M62M 60
11fd: PqIs A0fe b6!f &cFN 6A7D 2YC1 81
120f: hG7y EZqQ qurM !5ff nq!L K6c8 da
1221: BNSJ BykN 41Tb jg3W CBEw nYXV 35
1233: zxu& dqA5 xqSy 4X4m Ka!B H5bt 4d
1245: cqPV zWTd J#U% 504r xBc3 vfcG 51
1257: !gWM 2aB4 &8eh j74s !geM 2IzI 86
1269: w2h5 SgZj 6WB1 hbcE 6UQE wUk4 f4
127b: I22F 4Qgt kzzJ 5!k7 I0sA 4SPg 54
128d: 5aB2 23HK Uqlc INMF h4E5 A0mF bc
129f: gUQs e!2M 6WT! mvfc HuUt !k7g 31
12b1: 4ALM Cf1z 6m3# !ff% iriP 0Pwb 27
12c3: n1x9 NGUQ HvBb 4y%T Y2E2 yhlP 0a
12d5: D66M 174A NGdU hdsb Gk&I yMpN 9a
12e7: IQPW 74NL 7qTp !gag iqTN 0!Al e9
12f9: !h7g gaPW 6Yz8 z4LX 2Bn0 Ar06 ac
130b: PH1c dNSI q1S8 3sG0 RXaH ClX5 bl
131d: Hm9k 00C0 !kb% zms3 Qgb1 D1GJ 0f
132f: Xd59 PKqd U6aJ Xt41 Q5Fs cL1a 61
1341: xgaF 63zB 0ET0 bqQ1 TcBp Q5kI 3f
1353: 1H4& 76A& 0g7X 6!31 5L0j z6P8 16
1365: !IG2 8EJ1 1gGF #VsH ufel 0s80 e3
1377: ceXR Hd03 XLmI tu4w djqQ ncSa 7c
1389: wJ3X zMkU Hucz Gw4M 1ESX 7kMI 3e
139b: RR&9 7a8k xxk1 9bdp 3d04 PBxQ 16
13ad: 1UQS 7KVp xqRX !jqM Pgxt m&wu 81
13bf: 5ybI CFhw !Fxc qaTe Y0pY 5c9Q a7
13d1: LcKF p&Rn Kqxt sHPw W8Vg N#Up 68
13e3: Q4N# Wxs0 KlJE 9#NY Bqea 67DE 66
13f5: HVe5 EF04 VG7C ETmJ Gbwu 82Uv 5b
1407: jdQv HmZN Hshh !TIx Mf%M 2K2I d4
1419: twq& ahYZ PtLI PI98 vFqy 18Tr c0
142b: 7Ajl LfQM fcz# 8GQQ rJ07 LkL9 b6
143d: 590i jjHk y53R 0yxc 9N9K h%ws 83
144f: 1PD8 Gkz& 8a1g 8h5N L600 07kG d6
1461: pJjX Lz%U a90j uaXa #irN Q3N& 60
1473: f2gx wxdL URfg 5UM2 mUVf GBWR 2a
1485: HzYq YDT% g4rg 5rcj BW6g Ewg8 ca
1497: EsB7 Y179 id0F d2S3 ea3& NiaL a2
14a9: TC2w a8x2 yAyK kRyk mG83 8f8m 62
14bb: qaHp 123Z w4MW 7WBb IkHW Aqaa d0
14cd: D8xu 29db 0dMF jSHU Hqfg YWWj f4
14df: 7EUH 7W!k 7EMI 7!21 7BfW LH6w e5
14f1: !krM 2sB9 Y0kX d#VR aWAL 7Pd# f5
1503: za0p ul&K Y6SF 18nw j31I Y4Z4 f4
1515: Yu!j d4PS t4wP z125 Varx Rdqk 0c
1527: V8rz yJ84 VeSR 6k&Q ytjV m#7g 01
1539: x5%y e8Vn WuvN Wgug Bpc# xue5 17
154b: Uq90 xK3E xKaF HhC# 1e8W gudT 0c
155d: zmw2 U4SB Ug0t VK3C UGny !mxE ca
156f: rqBw 8e#s mI2X xaA7 EU6y 0mcA fc
```

```
1581: 6yex Dilm Fq0U WixK ZX02 NG5d 2a
1593: 6Nmg VzMV vUQd DrbS b4SF TN7L 95
15a5: cEQi N8NR E1Sq t&KF 420S bMeh fe
15b7: 8dpc jhMm MEUn cgEt Q8Ug qi!d ab
15c9: #6ie 2EQD u182 zing Wp#d 9H7a fd
15db: XzsD ux#t gaes ca8m rAlF q4E3 47
15ed: zmw9 c8RL KGAv zpLh d1Z6 zuP4 b5
15ff: zkUI 6w3M %2uN jpTM 13C5 1pQ0 f9
1611: 1FTE 1Kzg Yq8D 8eTN Ftgx skPT c2
1623: WaB0 zg0N ZKMt Gqyd 0&x5 XhSe 5b
1635: #psa #xKe v26y 7UX8 JT2d RIik 9b
1647: 7FSD 1cHg #IWp 1cX9 5L0R j7Ix 37
1659: HrdZ zP0E PyAy k979 DUy% XyGx a8
166b: 1aTW Q0fe c&Le 0xKJ dcDv Y05w ba
167d: Gv#d 0NKi C!Yu 51Aq Q8Qp Q21Y 15
168f: CdSz %qAf zhzk Gsud 1d!F j8Q5 e7
16a1: T5yF !8Qm Qa8a LsEm Y7#r 5MqF c4
16b3: 0pQn Syjb YaCg xsiB Nd3Y 84Qy 02
16c5: j5wq Ew1w #wUH mhxF 3Gig 0Krn c6
16d7: %03W 4d6w A0DM 0H0b !c06 Af3E fe
16e9: U0Gg S61U E7qc #CkQ G0h& E0mV 79
16fb: 404k H8jU 5x0s vqAJ 4s3E zA9V 83
170d: GjiN k&Ww 6aaF na0z 8bgm Ew!1 97
171f: bgn6 !x3W Gg2d &Ek! GiHw 5aat 97
1731: Rwsw Vf%M #YAd Y2P9 5d0b DqYn 17
1743: XIUx FYAw Aej9 &b3w J4Lw 3r3p a1
1755: DliF aj#t RgvK ma9c !iaw 3aU# b3
1767: 0YGZ Y5Sw LvEn xq6F 8aaS y13X cc
1779: HF9e Y0P8 K&03 Aq38 P7Y3 Aflw 8c
178b: 1pfi 5Q0w fzUw gB91 mq&x 7QYw 7e
179d: Nqdj 121e jRtp 8595 iQZi h24Y 18
17af: f00d 3glg jQh1 iy1k 1QZa hi19 c1
17c1: jAB3 iA4B mjEw 0203 44MN WGAx b1
17d3: E1Gd 50ec 5gdw %HX8 !cyF 08kg 81
17e5: FLIM 3grW ay&g NLK8 QfaG 6618 d3
17f7: Ev#5 #G87 xLLA %Wr# Q0b6 %Yr# 36
1809: q93t Uevg SqAT xg5& j20m Gh3C 58
181b: 4u&h xgiG Ldw7 82g7 FwhZ Kwt8 ca
182d: Fh1Z NMuE qa&2 Q0z0 0d04 !hrM c4
183f: S1xB %aG& pvS5 iqgh Y22a eekh 57
1851: I0f6 ijy5 ianY Vh6M 0IrZ xv!N f3
1863: i8yh %d3V N1bM yb58 NLT6 is&i 7b
1875: 4eQ0 21wU ufxU u00g 8420 0000 af
1887: 0000 0g80 0000 0043 1Ns3 10k6 a7
1899: 1Ms8 20g4 1g&7 2gEc 2EvU 9QJ9 0d
18ab: ji14 j45t dFd2 ikkM srta hlFg 33
18bd: gy13 i59p kRhl kSU7 1AQW EFxv fe
18cf: %%T% LPGG Z%#9 DrX% x!ft 5XDp 74
18e1: 5VAs 1Yzg Z!0z 1%16 82c7 Q30w b7
18f3: 8wtF 0IA4 A2vg 1!0z 1SA4 Q1Uw d8
1905: 8gtF 1IAd Q178 8247 qgT9 5d07 7a
1917: E0kw 90tF 7#U0 1eG5 4qr# FvYw 70
1929: ywuB i&n% Fky5 %y0z 1Uk2 Y1ga a7
193b: baA3 xh4w 8guC 0J02 qgww &gvM 21
194d: Cy0z 1%3F 8287 qgj9 1F3y Q0sw ef
195f: 8wtF 1J3p E0kw 90tF 2IAj Y0T9 06
1971: 5J3a E04w 90tF bd31 E0Aw 90uC 5d
1983: 48&i Ab&0 0000 0200 0000 0000 1c
```

```
0 REM ********************
1 REM * BASIC-PROTECTOR *
2 REM * (W)1992 POLONUS *
3 REM ********************
4 :
10 FOR T=49152 TO 49363:READ A$
20 GOSUB 50:POKE T,A:B=B+A:NEXT T
30 IF B<<>>19923 THEN PRINT "ZLE DANE!":STOP
40 PRINT"START PROGRAMU: SYS49152":END
50 B$=LEFT$(A$,1):GOSUB 70:A=C*16
60 B$=RIGHT$(A$,1):GOSUB 70:A=A+C:RETURN
70 C=ASC(B$):IF C>>64 THEN C=C-55:RETURN
80 C=C-48:RETURN
90 :
1000 DATA A5,2D,8D,B3,C0,A5,2E,8D,B5,C0
1001 DATA C9,09,B0,23,A9,1B,A0,C0,20,1E
1002 DATA AB,A2,FB,9A,4C,74,A4,0D,0D,50
1003 DATA 52,4F,47,52,41,4D,20,5A,41,20
1004 DATA 4B,52,4F,54,4B,49,21,0D,00,A0
1005 DATA 00,B9,00,08,91,2D,C8,C0,80,90
1006 DATA F6,98,18,65,2D,85,2D,90,02,E6
1007 DATA 2E,A2,70,BD,73,C0,9D,01,08,CA
1008 DATA 10,F7,A9,59,A0,C0,4C,12,C0,0D
1009 DATA 0D,50,52,4F,47,52,41,4D,20,5A
1010 DATA 41,42,45,5A,50,49,45,43,5A,4F
1011 DATA 4E,59,2E,0D,00,1A,08,C8,07,9E
1012 DATA 32,30,37,36,20,43,4F,4D,4D,4F
1013 DATA 44,B0,45,20,4B,45,42,41,42,00
1014 DATA 00,00,A2,80,BD,00,08,9D,80,03
1015 DATA CA,10,F7,4C,AA,03,A0,80,C6,2E
1016 DATA B1,2D,99,80,07,C8,D0,F8,A9,95
1017 DATA 8D,28,03,A9,FD,8D,29,03,A9,00
1018 DATA A2,00,85,2D,86,2E,20,59,A6,4C
1019 DATA AE,A7,4B,45,42,41,42,2D,42,41
1020 DATA 53,49,43,2D,53,54,41,52,54,45
1021 DATA 52,00

READY.
```

Witamy Waś Drodzy Czytelnicy w kolejnym numerze Kebaba. Mamy nadzieję, że numer ten otrzymacie punktualnie, to znaczy w miesiąc od ostatniego.

Jak już zapewne zorientowaliście się, numer który trzymacie zasadniczo różni się od poprzednich - wprowadziliśmy nową oprawę graficzną, oraz ponownie zwiększyliśmy objętość - tym razem już do 40 stron i będziemy starali się taką utrzymać.

W związku z tym postanowiliśmy sięgnąć do korzeni, czyli zbliżyć sie do tego skąd pochodzimy. Osoby będące z nami od początku doskonale wiedzą co mamy na myśli - oczywiście chodzi o scenę. Na tych dodatkowych stronach chcemy zamieszczać informacje o polskich grupach komputerowych, nowych demach itp... Już od przyszłego numeru planujemy stworzyć rubrykę w której będziemy przedstawiać polskie "teamy" zajmujące się pisaniem demek i nie tylko.

Jeżeli więc chcecie przeczytać na naszych łamach o sobie przyślijcie dokładne informacje o tym jak doszlo do powstania Waszej grupy, jej historię, aktualny skład, dorobek, to czym się aktualnie zajmujecie, adres, ewentualnie grupowe zdjecie. Mamy nadzieję stworzyć w ten sposób bank danych, który umożliwi Wam kontakty.

W numerze tym znajdziecie ankietę. Bardzo prosimy: nie omijajcie tej strony. Wytnijcie ją, wypełnijcie i przyślijcie na adres redakcji.

Pozwoli to dopasować zawartość merytoryczną pisma do Waszych gustów.

Równocześnie dziękujemy wszystkim za listy, i czekamy na dalsze...

Redakcja.



Uwaga! Ważny komunikat dla Czytelników, zainteresowanych KEBAB-MON-em (ogłoszenie poniżej).

Niestety z powodów technicznych musimy zrezygnować z taśmy jako nośnika programów. Pozostała jedynie dyskietka.

Również zmianie uległa cena i wynosi obecnie 80.000 zł. dla osób, które wpłacą po 20-szym października br.



Kupon ogłoszeniowy

....
imię i nazwisko

....
adres

treść:

....

....

....

....

